The Report Committee for Hans L Yeager
Certifies that this is the approved version of the following report:

Microprocessor Power Management and a Stand-alone Benchmarking Application for Android Based Platforms

APPROVED BY

SUPERVISING COMMITTEE:

Adnan Aziz, Supervisor

Andreas Gerstlauer

Microprocessor Power Management and a Stand-alone Benchmarking Application for Android Based Platforms

by

Hans L Yeager, BSEE

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2011

Dedicated to my wife, and best friend, Lesa.

Acknowledgments

There are so many people who have invested in my life, without them I would not be here today. My wife who has supported me through years of school and broken bones – without your love, patience and encouragement I would be lost. My father who introduced me to all of the hobbies and interests I have today – from the binary arithmetic to photography. My mother-in-law who taught me so much about Christ, my Lord. To my many professors who took extra time out to answer my numerous questions. To Dr. Adnan Aziz who guided me through the process of writing the Android application included in this report.

Microprocessor Power Management and a Stand-alone Benchmarking Application for Android Based Platforms

Hans L Yeager, MSE
The University of Texas at Austin, 2011

Supervisor: Adnan Aziz

Components used in mobile hand-held devices (smart phones and tablets) vary greatly in performance and power consumption. The microprocessors used in these devices also have vastly different capabilities and manufacturing limitations leading to significant variation effects. Battery life is a significant concern to the end users of these products. A stand-alone Android application capable of benchmarking a device's performance and power consumption is introduced. The application does not require the end user to have any analytic equipment or to have a technical background. This enables individual end users to better understand their particular device's performance and battery life interaction. They may also use the application to determine if their device's performance or battery life has degraded over time. Data is also uploaded to a central location so that devices can be compared against each other. The benchmarking application is capable of resolving variation effects caused by device, environmental changes and power management actions. This

application demonstrates the feasibility of creating a low cost ecosystem where thousands of devices can be quantitatively compared.

Table of Contents

List of Figures

Chapter 1

Introduction

Modern computing platforms are designed with robust power management systems in order to maximize performance and minimize power consumption. These power management systems must have several different power and performance levels available and they must possess enough intelligence to correctly use the many options. Improper selection of a power and performance level results in performance loss and/or excess power consumption. Both hardware (HW) and software (SW) play a significant role in these power management systems. For example if the end user is listening to music, but not using the display, the battery life may be extended by turning the display off extending the amount of time the end user will be able to listen to their music.

Virtually all micro-processor (and computer system) design teams are facing this issue of power and performance optimization. It does not matter if they're working on processors for hand-held smart phones consuming just a few mW[5] or processors for large multi-node machines consuming several KW[16]. Transistor scaling continues enabling design teams to put more transistors in the same area with each new process technology node. However voltage and

frequency are no longer scaling according to classical trends. This is leading to ever increasing power densities which are stressing existing cooling solutions and will likely lead to advanced liquid cooling methods especially as 3D chip integration becomes more common over the next few years[56]. Processors for hand-held mobile applications and high performance computing systems have completely different power, thermal and area constraint magnitudes - but they generally have the same set of constraints. Thus, the design teams working on these systems must address the same power management issues. How to provide the desired performance (which is time-variant depending on the end-user demand) while minimizing power consumption. Additionally, the desired performance must be achieved while ensuring the processor does not fail or wear out prematurely.

The number of unique and dedicated hardware sub-systems (audio decode, graphics, video encode, video decode, memory and IO-subsystems) being added to mobile platforms is large. Manufactures are specifying performance metrics for usage scenarios on data sheets because end users value them. Examples include:

- phone talk time and stand-by time
- web-browsing speed and time
- digital audio or video playback time
- game play time

Time (specifically the amount of time a task can be performed on a single charge of the battery in the system) is a key factor in all of these performance metrics. As a result various system components are shutdown when not in use to save power. The significance of these factors to the customer cannot be denied and is further evidenced by the amount of discussion and articles on these topics found on the internet. Examples include articles giving tips to reduce the device's battery consumption rate[7], and blogs where people share device battery life problems and potential solutions[6].

Thus, there is a need to determine how a device's battery life time depends on the usage scenario. This has led to the development of mobile platform benchmarks (for example Mobile Mark) but these are usually proprietary and must be run in a lab setting to achieve sufficient control to post the results in accordance with the benchmark developers. Furthermore, when a web site (such as `www.anandtech.com` or `www.cnet.com`) does a comparison of devices, they typically make measurements on a single unit for each device type. Unfortunately, Silicon components have significant manufacturing variations. These lead to performance and power variations making it difficult to accurately compare different device models. Also, some battery technologies have a tendency to have noticeable changes in the charge that they can hold over time. These are just a couple of factors that indicate it would be beneficial to consumers if there was a way to collect power and performance data on thousands of mobile devices with the ability to compare them against each other. It would also be beneficial if a user could benchmark their own device

for performance and power consumption - allowing them the ability to check it again at a later time to see if their system had degraded. It would be ideal if a stand-alone application could accomplish this because it would not require the user to have any additional HW (or a technical background). The user would simply download the benchmarking application, run it and upload the data.

This report introduces an Android application capable of benchmarking the host device's performance and battery life. The data may then be uploaded to a central database hosted by a Google Appengine application. Chapter 2 introduces this Android application. Parameters directly controlled and monitored are described as well as the User Interface (UI). Chapter 3 discusses several technical issues that were solved during the application development. Chapter 4 reviews data collected during beta testing by several people using a wide range of devices. The data analysis assumes the reader is very familiar with the background material covered in the Appendices. In particular, the expected trends are used to explain some of the differences in measured data points that would otherwise appear to be random variations. Finally a conclusion is given in Chapter 5 which includes an outline of future work required to take the proof of concept presented in this document to production.

Chapter 2

MobilePowerBench: Mechanics and UI

2.1 Project Vision

One key objective of this project is to demonstrate it's possible to collect large quantities of useful data from many different devices, enough data that device variation can be observed. A stand-alone application capable of benchmarking each device is needed to keep costs down. Leveraging the open-source philosophy, a free application might entice end users to contribute the power and performance data from their individual devices to a central database. If enough end users participated then they'd all reap the rewards of a large quantity of data upon which to base future purchasing decisions. The application would also enable each end user to qualitatively track their device's performance over time, detecting signs of fatigue or rouge processes running the battery down.

Power consumption may be bounded according to virus, idle and minimum activity test cases (Appendix B.1). Real world usage scenarios (web browsing, audio playback, video playback, YouTube, etc.) can be correlated to system component virus, idle and minimum activity power values (Appendix B.1.5). The overall battery life and performance is dominated by certain com-

ponents in the platform. The variation of components dominating battery life also dominate the platform's battery life variation. For example when the screen is on it is the highest power consuming component in the system and the variation in screen brightness dominates the battery life. When the screen is off the silicon components dominate power consumption and variation in battery life. Another key objective of this project is to show it's possible to obtain insight about the power consumption and power management of these mobile platforms through the use of well crafted test cases in combination with controlling (or measuring) the dominant platform components.

2.1.1 Proof of Concept Goals

The MobilePowerBench application seeks to demonstrate it is possible to:

- enable users to determine if their device's performance shifts over time
- measure power and performance of the device under different scenarios
- find upper and lower limits of battery life for interesting scenarios
- compare the power and performance data across different devices and different device models
- work completely independent of any analytic lab equipment
- run on a wide range of Android devices

- be downloaded and run by the general (nontechnical) public
- provide a method for end users to upload their data anonymously
- collect data on non-controlled variables for correlation purposes

2.1.2 Future Goals

Demonstrating proof of concept should make it possible to:

- compare unit to unit variation and provide end users with some insight as to whether or not they have a particularly bad (or good) device
- gain insight about power management interactions between hardware and software which can be used for research
- use the MobilePowerBench application as a tool to evaluate how changes impact the overall power and performance capabilities of these devices

2.2 Application Overview

2.2.1 Basic Operation

Conceptually, the application is quite simple - select a benchmark and run it. When the benchmark finishes all results, new and old, for that benchmark are displayed on a scalable graph. The user may choose to upload the data to a central database with the touch of a button. There are a limited number of power management and performance settings that may be adjusted,

but default settings are provided. The rest of section 2.2 describes various parameters captured by the application. Section 2.3 covers the user interface in detail.

2.2.2 Parameters Under User Control

There are a few parameters the user can directly control that have first order impact on battery life and performance. These are adjusted by the user on the "settings" screen. First the Android API for power management (PowerManager.WakeLock) setting is employed to control screen brightness, keyboard lighting, and CPU power on status. The following power management settings can be set/changed by the user:

- cpu-screen-keyboard = "on-bright-on"
  - CPU is forced to stay powered on even when the workload is low
  - the screen is forced to max brightness at all times
  - the keyboard is force to be on at all times
- cpu-screen-keyboard = "on-bright-off": same as on-bright-on except the keyboard shuts off after the time out period
- cpu-screen-keyboard = "on-dim-off": same as on-bright-off except the screen changes to a dim setting after the time out period
- cpu-screen-keyboard = "on-off-off": same as on-dim-off except the screen powers down when not in use

- cpu-screen-keyboard = "none": no forced power management, the API is not called or used

For the CPU specific benchmarks, the application also enables the user to run either one or two threads simultaneously and also set the thread level priority for the CPU benchmarks. When entering the run activity screen, the thread (priority and number of threads to use) settings are automatically set to "1" unless one of the following CPU intensive tests has been selected: cpu-float, sort, sort tiny.

This is a key capability as devices now have at least dual CPU cores on the SoC (like Nvidia's Tegra 2 SoC found in both the Samsung Galaxy Tab and the Motorola Xoom).

2.2.3 Monitored Parameters, NOT Under Direct User Control

The MobilePowerBench application does NOT collect any user data or unique device identifiers (*i.e.*, a serial number). The only way to know that some data points are from a single device is to save multiple data points in a single data record. When the data record is uploaded, all the data points contained in the data record are uploaded together. To accomplish this and enable the user to take multiple data points with less effort, the user may specify the percent of battery life over which to make measurements each time the application is run. The parameters that are captured with each battery life status update are:

- time (used to compute elapsed time)
- performance metrics (frames rendered or computations completed)
- CPU utilization and number of running processes
- battery level, voltage, temperature, status
- if there is an active internet connection

Up to six measurements (at battery level status changes) may be captured per data record. This enables the differences between values at any two measurement to be used to calculate the performance, time and battery life changes for up to five useful data points per data record. At least two "good" measurements, corresponding to one data point, must be made in order for the record to be saved.

It's important to note that the end user may turn on many applications, or specifically shut them off, through the normal Android mechanisms. The user directly controls these parameters and control is not provided through the MobilePowerBench application itself. If two or more identical devices have different performance and/or battery lifetime it is expected that the additional information being measured will help explain the observed differences. For example if one device has 50 active processes but the other device only has 20 then it is expected that the device with 50 active processes has lower battery life. These parameters help explain differences in measurements even though the parameters are not directly controlled. This helps isolate variation effects

that are not caused by specific unit characteristics or caused by measurement variation effects (see Section 3.6).

2.2.4 Static Platform Parameters

In addition there are some platform, hardware and software parameters which are captured with each data record. These are time invariant, and so they're captured one time for each record:

manufacturer	board name	brand
Android Version	Android API level	model name
display name	CPU instruction set	device name
software fingerprint		

2.3 User Interface (UI) and Operation

2.3.1 Intro to the Four Activity Screens

The functions required by the application are partitioned into four unique activity screens. This partitioning is used to ensure buttons and information are properly displayed on very low resolution displays. The partitioning also separates the settings and run screens to ensure that, while running, a user can not change the settings. There are other mechanisms that could have been employed, however they have problems. One example requires the user inputs to be ignored while the test is running. This creates user confusion for subsequent runs as the internal values contained in variables are not be in sync with what is currently being displayed on the screen. The four activity screens are: Main (top level), Settings, Run, Instructions.

This partitioning requires passing data objects between the activities in the application - not a straight forward proposition. The final implementation passes complex data objects between activities using singleton classes. Exactly one instance of an object for a singleton class can exist in an application. Any activity screen may call the constructor of a singleton class and the constructor returns a reference to the (one) instance in existence, effectively passing the complex object between activity screens.

All of the screens have two common UI features:

- The upper left button is used to exit the active screen and go back to the previous screen. This is called the *home* button in the Settings, Run and Instr screens and it takes the user back to the Main screen. On the Main screen this is the *quit* button and exits the application.

- The upper right button is a *help* toggle button. When the *help* button is toggled on, other buttons may be pressed in order to obtain a help message explaining the button's usage. This provides a context sensitive help capability in a compact form.

2.3.2 Main (top level) Screen

The Main Screen (shown in Figure 2.1) is dominated by a 2D scatter plot of all data points, contained in all records, associated with the selected test. At startup a default pattern is displayed. A Chart Engine, `www.achartengine.org`, was used under Apache 2.0 license to display the data. This enables touch gestures so that the end user may zoom in/out and shift axis to see both global trends in the data as well as accurately see specific measurement points. Because the battery does not change in emulation mode, a photo of the main screen for a real device, displaying real measurement data, is shown in Figure 2.2.

- *select test* spinner enabling the user to select the active test. This also impacts:
  - the data displayed on the 2D scatter plot
  - the instructions that will be displayed if the *instr* button is pressed
  - the test that will be run from the Run screen if the *run bench* button is pressed
- *upload* button used to upload data records to the Google App Engine database.
- *run bench*, *instr*, *settings* buttons take the user to the other three screens

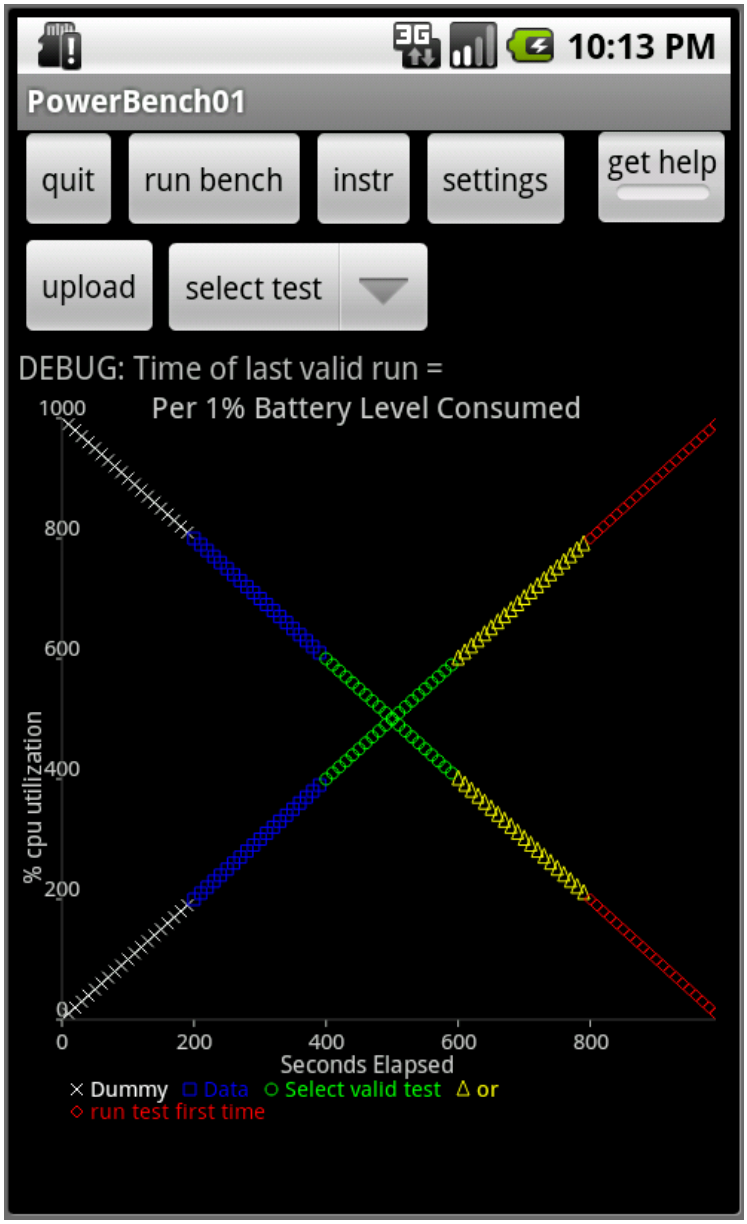

Figure 2.1: Main Screen Displayed at Startup

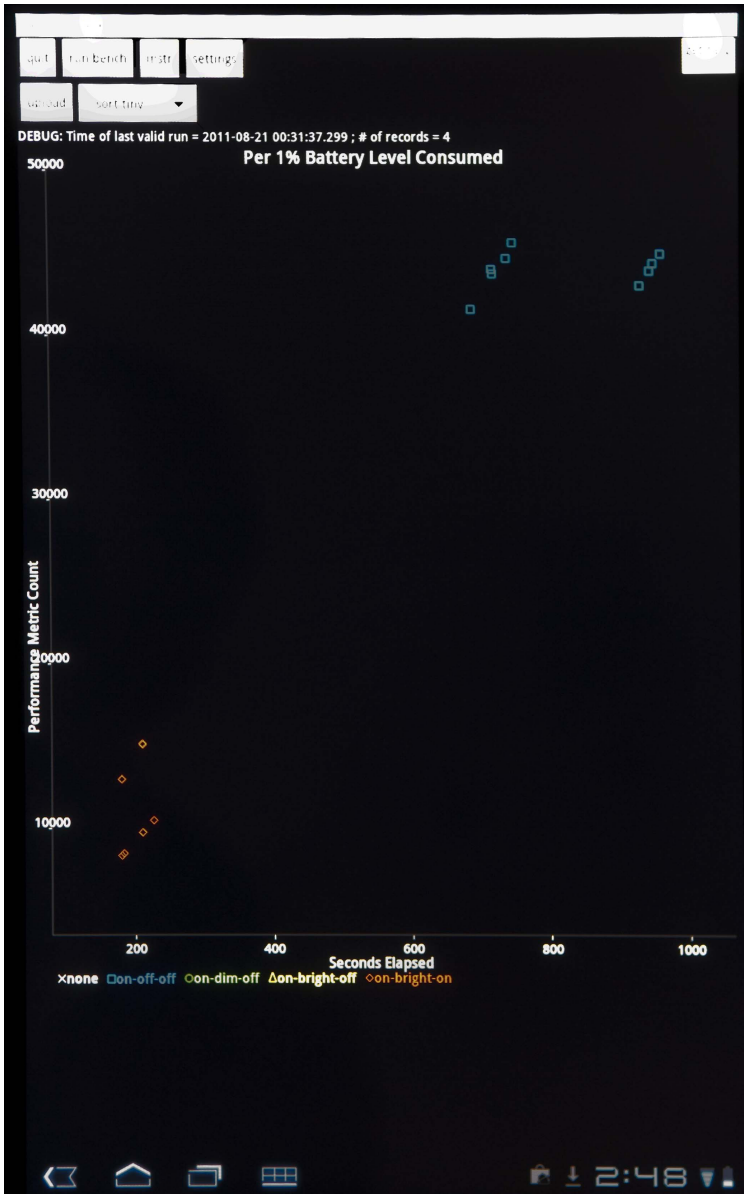

Figure 2.2: Main Screen Display for a Real Device and Data Collected

2.3.3 Settings Screen

The Settings Screen facilitates user input for things they can directly control through the application described in section 2.2.2. Figures 2.3 and 2.4 show the Settings Screen with different selections.

- *power management* spinner used to set the screen, keyboard and CPU wakelock
- *number of threads* toggle button used to select the number of threads (one or two) launched if doing CPU intensive benchmarks
- *thread priority* spinner used to set the thread priority (from 1 to 10) if doing CPU intensive benchmarks
- *percent battery level* spinner used to select the number of data points to collect over a single run and saved in a single data record

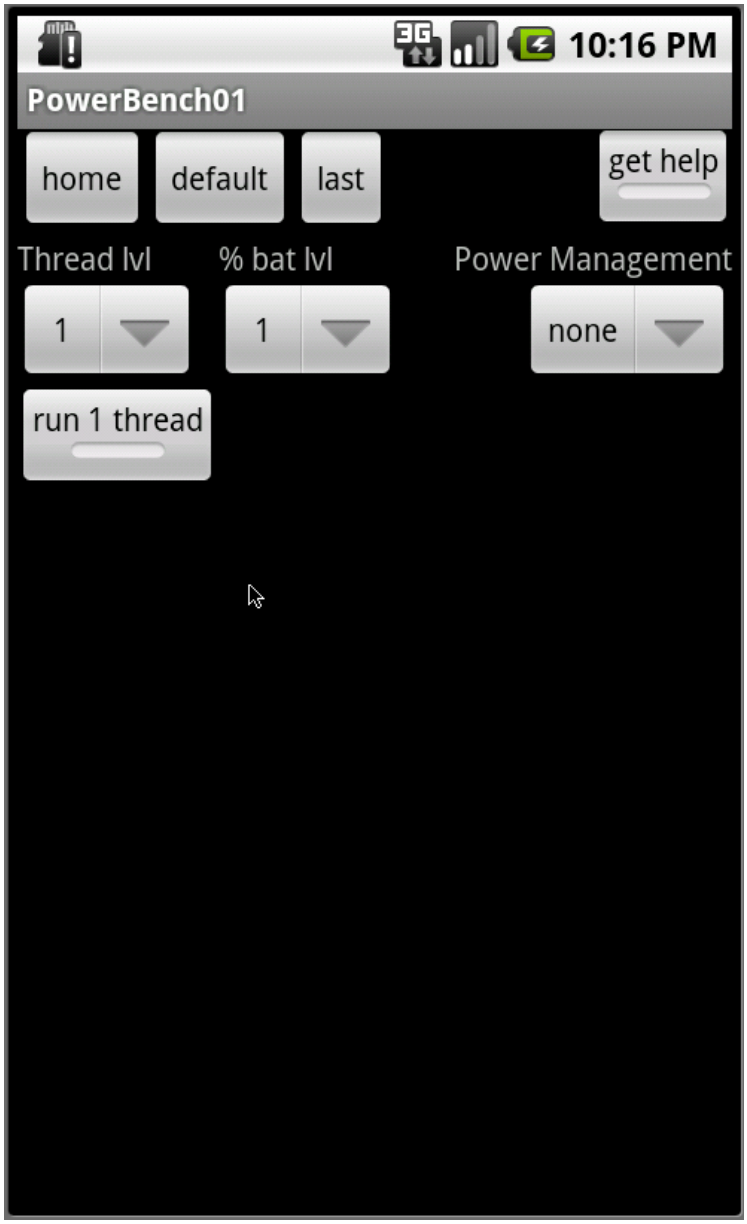

Figure 2.3: Settings Screen Defaults

Figure 2.4: Settings Screen with Changes

2.3.4 Run Screen

The Run Screen is used to *start* or *cancel* a run of a benchmark (Figure 2.5). Before changing to the Run Screen the benchmark to be run is selected from the Main Screen and other settings are chosen from the Settings Screen.

- Displays the status of the application that's actively running.
- For the 3D pyramid (graphics) test, the pyramid will be displayed on the screen.
- Note: the start button is disabled after it is pressed one time unless the cancel button is pressed to cancel the run. This prevents multiple threads from being started.
- Note: the home button is disabled to keep the user on this screen for the entire benchmark run.

Figure 2.5: Run Screen for 3D Pyramid test selection, test measurements have not been started

2.3.5 Instr(uctions) Screen

- This screen provides different information depending on the test that has been selected on the Main Screen. Specific instructions for each unique test are displayed (Figure 2.6).

- When the Instr screen is activated and no test was selected on the Main Screen the full application instructions are displayed (Figure 2.7).

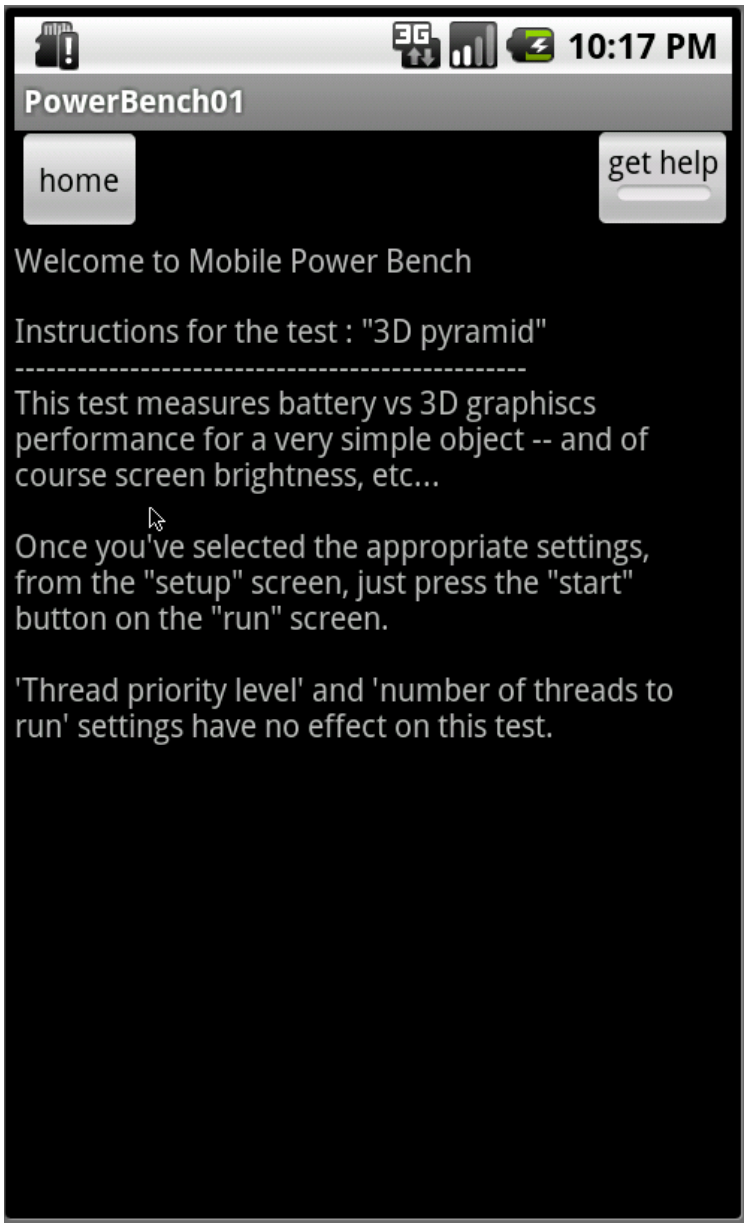

Figure 2.6: Instr Screen for 3D Pyramid test selection

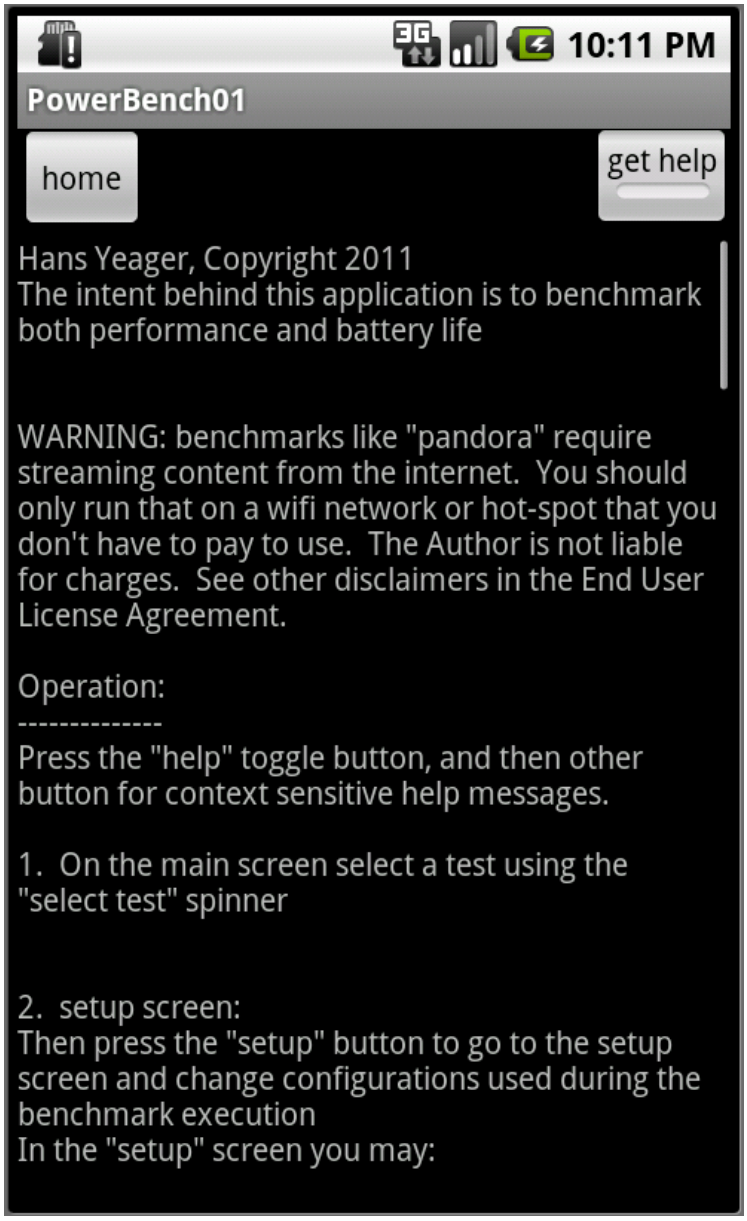

Figure 2.7: Instr Screen for no test selection - screen will scroll to show full instructions for the application.

2.3.6 Making Measurements

The Android battery status listener is the mechanism used to trigger a measurement. Every time the battery status changes the operating system is directed to call a function in the MobilePowerBench application. There are several things that can cause the battery status to change (and thus trigger the code). For example:

- battery level (percent consumption) changes
- the device is plugged in (or unplugged)
- battery temperature changes
- battery voltage changes

Each time the battery level updates, the parameters being monitored are sampled and stored. By sampling the parameters being monitored at the specific transitions of the battery level, it is possible to ensure that all data points are normalized to a certain percent of battery life consumed. This is a powerful normalization. End users are not concerned with how many Joules a particular operation consumed, rather they want to know how long they're going to be able to use their device before it must be plugged in again. By normalizing the performance to a percentage of battery used (and the corresponding time elapsed) it's possible to predict device lifetime (between charges for specific benchmarks) without consuming the entire battery. This also provides a relevant metric through which vastly different platforms may

be compared - and still provide end users relevant data. The other advantage of this method is that it includes the impact of battery variation on battery life; some batteries will not hold as much charge, or operate as efficiently, as others.

To ensure the quality of the data collected, the application prevents a run from starting if the device is plugged in (to a USB or AC outlet). If, at any time, the battery status changes because the device was plugged in, an active run will be canceled. Usable measurements made prior to the device being plugged in may be saved, but the final measurement that would have been made if the device was not plugged in is thrown out. This ensures that all data points are collected while the device is running only on battery power.

Chapter 3

MobilePowerBench: Technical Challenges

3.1 Introduction

There are several challenges associated with an application of this nature. Some challenges are driven by business considerations, the desire to keep end-user data anonymous and the number of variables that should be controlled vs the cost and effort to control them. There are also technical challenges associated with using a system to make measurements of itself. This chapter reviews several development challenges, the decisions that were made and the motivation for those choices.

3.2 Anonymous Data Implications

It is highly desirable to avoid collecting personal data regarding people or devices that have uploaded information to the server. The reason is to avoid legal responsibility for protecting that data and ensuring it is only used for the purposes intended. As a result, the MobilePowerBench application does not collect any personal information nor does it require a login to upload data to the server. No accounts or passwords are required. While this solves a legal problem it creates another problem: it's not possible to determine whether

two data records came from a single device. If the data records collected are truly anonymous, it is impossible to determine later if they came from a unique device or not. The static platform parameters captured with each data record can help correlate different data records and suggest that two different data records might have come from a single device, but this is not a guarantee. However, multiple data points contained within a single data record are uploaded together. In this way it's possible to guarantee that some data points are correlated to a single device even though the data record was uploaded anonymously.

3.3 Control Conundrum

There is an interesting trade-off with respect to the amount of control that can or should be made while benchmarking the devices. This is a compromise between a precise scientific approach to the data collection, where all possible variables are controlled, and the desire to capture real-world data based on how people use their devices.

A scientific approach includes preconditioning the device, such as a clean installation of the operating system and the applications required for testing. The environment must also be controlled. For example, temperature has a strong impact on leakage power and the voltage required to achieve a certain frequency of operation (Appendix A). Analytic equipment is then used to measure the energy consumed for a given function such as audio playback[17]. A common approach is to loop on a benchmark test suite and measure the

time it takes to completely discharge the battery. This method produces repeatable measurements but each measurement requires several hours[4]. A lot of work is required to obtain data for just one or two systems.

In reality, some users never have more than one or two applications open at a time. Other users might have 10-20 applications open with many background tasks running. Actual data, collected during beta testing, showed that the number of active processes ranged from 22 to 70. Beta testing also revealed battery temperatures ranging from 24.0C to 45.5C. These are real world usage modes, yielding a different performance and battery life experience for the respective owners.

Precisely controlled measurements require expensive analytic equipment, complex setups, and tend to neglect variation of end-user conditions. It would be very expensive and time consuming for an independent company to make enough measurements, using a scientific approach, to cover the potential variation of both usage conditions and devices.

The MobilePowerBench approach leaves many variables uncontrolled but measured. This enables an application that is accessible to nontechnical end users. At the same time the application does measure many key parameters that impact the performance and battery life. The list of parameters is given in Section 2.2.3. The additional parameters are measured and collected at the same time as the performance and battery life measurements. These extra parameter measurements can be used to explain performance and battery life differences occurring between measurements having the same user con-

trolled input settings. In this way data can be collected over the wide range of conditions, by nontechnical users, without the aid of any analytic equipment. If enough data points are collected the additional parameters being monitored may help distinguish between variability caused by the device's HW/SW and variability induced by realistic differences in end-user environments.

3.4 Devices with Limited Battery Status Resolution

Some Android devices update the BatteryManager(EXTRA_LEVEL) status in 1% increments. Other Android devices update the battery level status in 10% increments. Systems updating in 10% increments require a 10x increase in the benchmark run time. If all other things are equal then a device updating the battery status in 10% increments should produce performance data points equivalent to the average of ten data points from the same device updating the battery status in 1% increments.

There is also a question of the accuracy of the hardware used to determine if 1% or 10% of the battery has been consumed. A device requiring battery status updates in 10% increments (one significant digit) does not need HW having the same tolerance as a device requiring battery status updates in 1% increments (two significant digits). A measurement taken between battery levels from 90% to 80% and a measurement taken from 20% to 10% may not collect data over the exact same amount of energy consumption. Multiple data points from different battery levels and different units are required to observe variation that may result from these limitations.

During beta testing, the Motorola Droid, Droid2 and DroidX uploaded data records having battery status updates in 10% increments. Users appear to be impatient above a certain amount of time that they must wait for the benchmark to complete because just 16 data points were obtained. A solution for this problem was not identified during application development. Other devices tested show battery status is being updated in 1% increments.

3.5 Devices with Voltage Measurement Inaccuracy

To further complicate matters, the battery voltage will drop as energy is removed from the battery. It is expected that the platform's energy consumption rate will change with time. Figures 3.1 and 3.2 show battery voltage vs percent battery life consumed. Note that the absolute voltage levels correspond to absolute battery level (not shown); this is why some data records with the same device name are at different absolute voltage levels. Also note that each data series comes from a unique data record corresponding to a single device. Thus each data point in a particular data record is known to be from the same device and to have occurred in consecutive order.

Notably the Samsung GT-P7510 is perfectly flat at 4 Volts. This is an outgrowth of how this particular device has implemented the Android BatteryManager(EXTRA_VOLTAGE) API - it only returns one decimal digit of precision (in Volts). As a result, the data for this device is always either 4 volts or 3 volts; this is not very useful. However, the other devices are providing four decimal digits of precision (in milli-Volts). The Motorola Xoom shows a clear

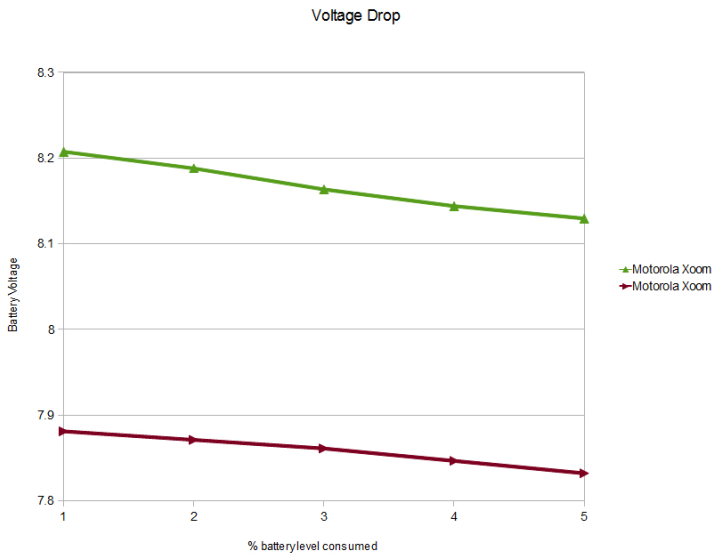

Figure 3.1: Voltage vs Consecutive Battery Life Steps

voltage drop as energy is consumed. It seems to have an accurate on board voltage sensor supplying information to the API as evidenced by the steady drop in voltage with each data point in the respective data records. Also note that the Samsung SCH-I500 device shows a downward trend of voltage, as energy is taken from the battery, for each data series. The Samsung SGH-T959V shows an overall downward voltage trend, comparing the voltage at 1% vs 5% battery life consumed, but some data points go up and then back down. This tends to indicate the device has less accuracy in the hardware being employed to determine the actual battery voltage.

At first glance, it might be tempting to conclude that there is too much inaccuracy in the data to be useful. However information about the hardware's

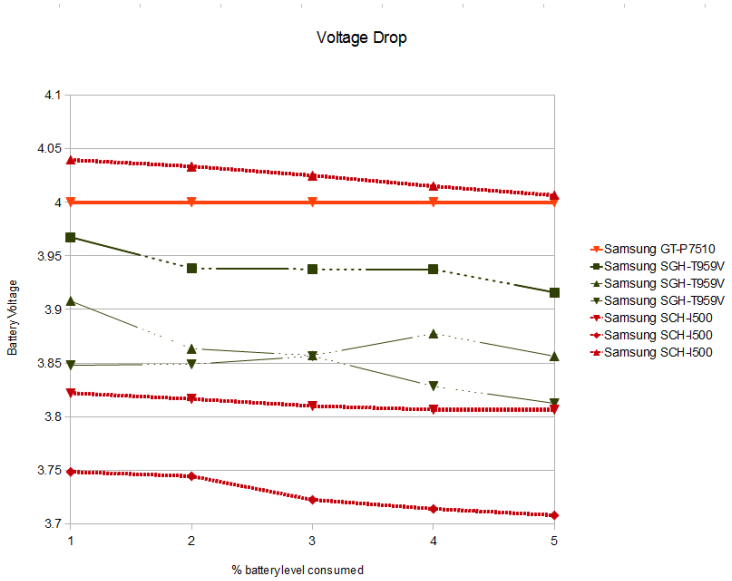

Figure 3.2: Voltage vs Consecutive Battery Life Steps

accuracy can be determined by post processing the data. By analyzing all five data points coming from a single data record the variation of each device can be determined and compared across multiple data records. This is an example of how multiple data points, contained within a single data record, can be used to work around the problems of uncertainty in the data measurements. This is also an example of how data may be purely anonymous yet still provide sufficient resolution to be useful for correlation purposes later.

3.6 Software Conflicts

Another critical issue associated with using the device to measure its own power and performance stems from the fact that the OS is responsible

for doing both and can run into conflicts. In particular, the OS is responsible for handling the hardware initiated battery status events used to trigger a measurement. The OS is also responsible for scheduling the execution of all other processes required by the kernel, other user applications, the MobilePowerBench application and benchmark threads. This can be of particular concern in a platform with a single processor core because the OS and user processes can not be running simultaneously. There will be a time lag between the hardware's assertion of an interrupt indicating a battery status change and when the interrupt is processed by the OS. Then the OS must figure out that a software routine in the MobilePowerBench code must be executed. The battery status change routine in the MobilePowerBench code must then be task switched in to execute on the CPU in order to finally make the desired measurements. The time lag will vary depending on several factors:

- the number of HW core processors in the platform
- the number of active software processes
- the priority of software processes
- the priority of event handling

There can be a significant lag in time if the CPU is busy with other tasks. During testing this effect was observed on a single CPU device when the cpu-float test was used with a single thread and maximum thread priority (of

10). The OS appears to have been giving CPU time preference to the benchmark thread, because of the high thread priority setting, delaying interrupt service for the battery status listener events. This resulted in a wide variation (11 seconds) in the amount of time (and performance) associated with each 1% drop in battery level as shown in Figure 3.3.

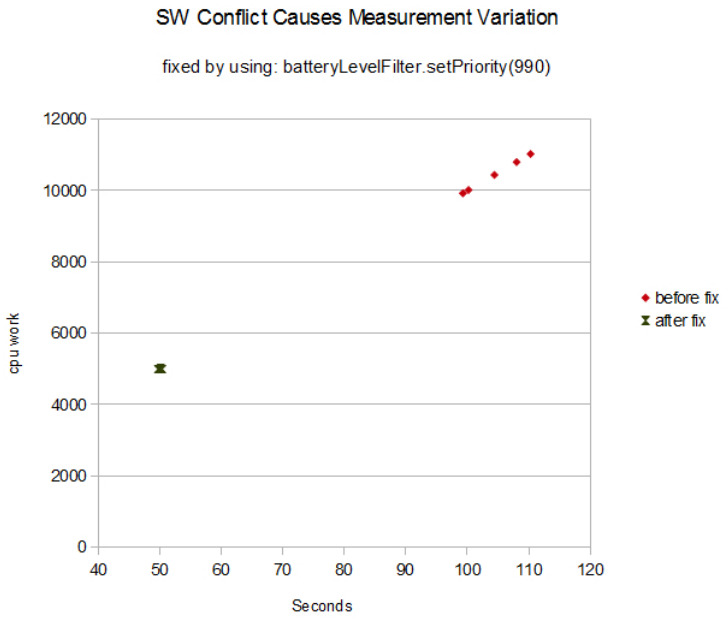

Figure 3.3: Measurement variation due to software conflicts between the benchmark thread priority and the battery status listener - before and after increasing the battery status listener's priority

Setting the priority of the battery status listener to 990 (1000 is reserved for the OS) with the command *batteryLevelFilter.setPriority(990)* corrected the problem - significantly reducing the variation between data points. After the fix was implemented all five data points, on this device, were within 0.7

seconds of each other. The data points collected after the fix were at a lower performance and time because the power management setting for the screen brightness had changed. However, the thread priority = 10, number of threads = 1, and test = cpu-float were the same for both data sets. Even with this improvement, there is still some measurement variation as a result of this SW conflict. Data analyzed in Chapter 4 indicates that the measurement variation is a function of the thread priority, number of threads, number of cores and SW version.

The key challenges facing the development of an application such as MobilePowerBench can be overcome through the use of packing multiple data points within each data record. Post processing the data points contained within a data record provides sufficient information to determine the impact of measurement variation effects. The consecutive data points in a data record may be combined to find their mean and the standard deviation could be used to quantify the uncertainty. Users are able to submit data anonymously avoiding costs associated with legal issues and a need to protect people's privacy. The costs of additional analytic equipment is avoided. Chapter 4 contains examples showing that the measurement of key parameters, instead of directly controlling those parameters, can explain differences in performance and battery life when the small number of user controlled inputs are identical.

Chapter 4

MobilePowerBench: Data Analysis

4.1 Data Overview

Most of the Figures presented in this chapter plot "performance" on the y-axis and "seconds" on the x-axis. There are a few exceptions and explanations for these are given in the text.

It is important to remember that all data measurements are normalized to 1% battery life according to the discussion in section 2.3.6. As a result, the "seconds" label on the x-axis represents the amount of time it took for the battery level to drop by 1%. So the x-axis represents seconds/energy where the amount of energy is normalized to be 1% of the battery capacity. This normalization facilitates a useful comparison of the different devices because end users are really interested in how much work they can get done before the battery is exhausted (on a single charge).

The term "performance" is used to be more intuitive to non-technical end users - specifically a higher performance rating is commonly understood to be desirable. However "performance" in this case really represents the amount of work accomplished by the benchmark. Specifically the cpu related benchmarks run in a loop and the loop counter is used as the "performance"

metric. When two threads are running in parallel, the sum of both loop counters is used as the "Performance" metric. The benchmark can only do work for an amount of time corresponding to 1% of the battery capacity, so the y-axis has units of work/energy.

4.2 cpu-float Test

The cpu-float test is extremely simple, consisting of data dependent add, multiply and divide operations in a loop. A small number of variables and constants are used to keep the memory footprint small. The small memory footprint reduces the overhead associated with task switching between the benchmark thread and other processes. A single cache line transfer is capable of restoring all the data required for the cpu-float thread to continue making computational progress. Thus the performance metric is minimally influenced by poor OS and kernel behaviors such as improper binding of the thread to specific processors.

In Section 3.6 the impact of thread settings on performance measurement was introduced. Next the analysis is expanded to evaluate how performance changes with different thread settings across several platforms. Figures 4.1, 4.2 and 4.3 show cpu-float performance vs time, per 1% battery life for three different models. The number of threads, thread priority, and power management settings have been varied. These particular devices have been selected for detailed analysis because of the relatively high number of data points collected for them; additional models will be discussed afterward.

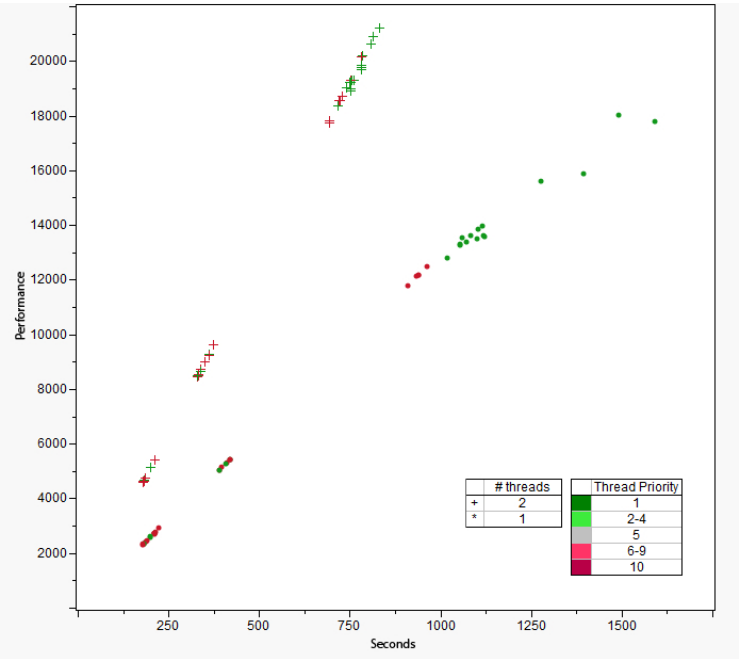

Figure 4.1: Samsung GT-P7510: cpu-float performance vs time for different thread settings

The GT-P7510 has modest measurement variation even for two threads and a thread priority = 10. The GT-P7510 is a dual-core processor running OS version 3.1. The single thread, thread priority one, variation is explained in Section 4.6 after considering some of the other parameters that are measured but not controlled (as introduced in Section 3.3). Interestingly, the Desire HD and SGH-T959V devices have significant variation, even for some single thread and low thread priority runs. Both of these devices are running Android OS version 2.2.1.

Several other key observations can be made from the GT-P7510 Figure

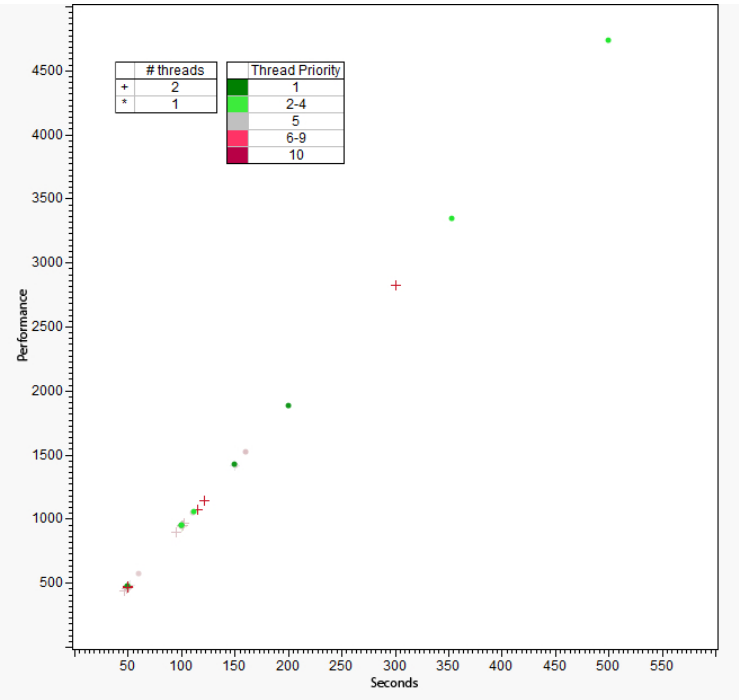

Figure 4.2: HTC Desire HD: cpu-float performance vs time for different thread settings

4.1. First, running two threads in parallel results in significantly higher performance than running a single thread. As expected, the performance does not double because there are other processes running that must consume processor time. This also implies that there is very little overhead or physical resource conflicts between the two threads. This is expected because the test only uses a small number of registers.

Second, there are three distinct regions of time that the data points fall into: (1) less than approximately 250 Seconds corresponding to the screen on max brightness, (2) a band in the 300-400 Second range corresponding to

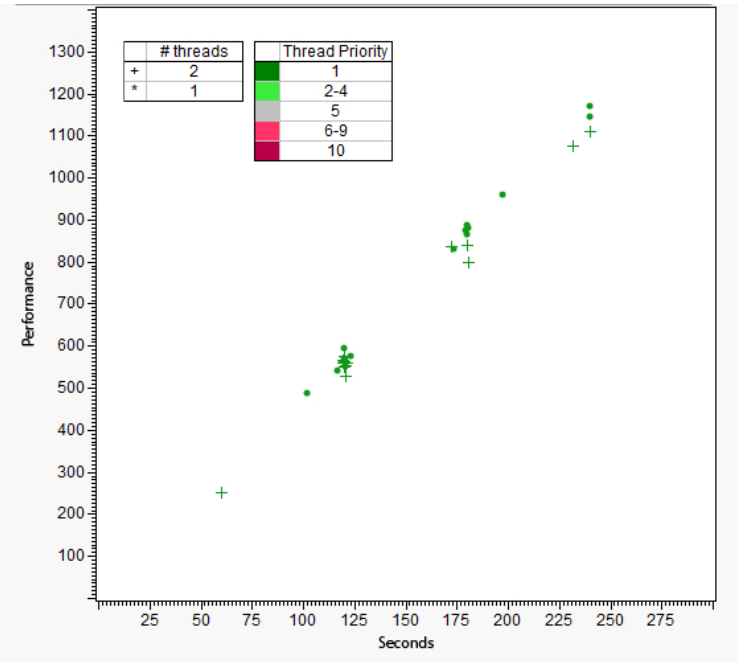

Figure 4.3: Samsung SGH-T959V: cpu-float performance vs time for different thread settings

the dim screen setting and (3) greater than approximately 600 Seconds corresponding to the screen off, but the CPU still on. In the region where the screen has shut off, it's observed that using a single thread does result in more time, per 1% of battery life. The CPU utilization is just over 50% (ranging from 50.02% to 50.08%) for the single thread cases. This implies that some power management techniques (Appendix C) are being employed to reduce the power of the core having low utilization. This suggests that the MobilePowerBench application is capable of detecting active power management techniques being employed by the system, even silicon level power management when the screen

is off.

Third, it's observed that a lower thread priority also results in longer battery life. This is another indication that the power management system is active and working to try and provide the desired performance level. However, it also implies that there may be some room for improvement because the high thread priority simply consumes 1% battery life earlier without achieving performance improvement over the low thread priority case. Looking at the CPU utilization data shows that the system and IO activities are slightly higher for the cases where the thread priority was 10 instead of cases where it was set to one. Additional debug and attempting some changes to the scheduler in the kernel could be explored. Nevertheless, this is a good indication that the MobilePowerBench application could be used to identify and help debug power management problems.

Figure 4.4 shows all device data for the cpu-float test where the colors represent different power management settings. Note that this still contains all of the data including various thread priority and number of thread settings.

The Motorola Xoom appears to have a large variation in the measurements - much more than the GT-P7510. Also note that the five SCH-I500 data points with a performance level in the 8,000 to 12,000 range are from the same data record indicating a significant variation in measurement. These five points should be averaged together. Some of the interesting trends that emerge from this data are listed below. The most important result obtained from this data is that it is possible, given a large enough sample size, to ascertain the

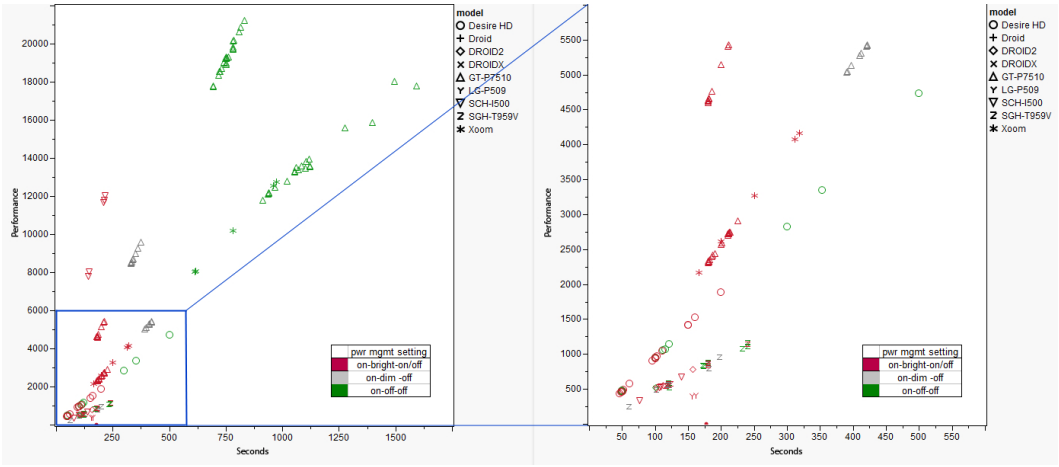

Figure 4.4: All Devices: cpu-float performance vs time for different power management settings

relative performance of the different devices.

First, the SCH-I500 has good performance even though it burns through its battery the fastest. It's notable that there are another five data points which have low performance. Both of these runs launched a single thread with thread priority one. Additional analysis of the data collected revealed that there is another significant user process running, along with MobilePowerBench, when the performance dropped. The low performance SCH-I500 data points, with performance in the same range as the SGH-T959V and Droids, had a CPU utilization at 100% while the high performance data points had a CPU utilization in the range of 40%.

Second, the GT-P7510 and Xoom are essentially on the same performance power curve. Both of these devices use the dual core Tegra 2 processor;

however the Xoom has higher measurement variation. This is interesting in that both systems are using Android OS version 3.1. Typically the Xoom only had about 23 active processes while the GT-P7510 had about 26. The Xoom data was only run with 1 thread at a time resulting in a CPU utilization around 50%. The Xoom and GT-P7510 do have different SW build fingerprints though. Perhaps something else in the firmware or other software is causing the high variation observed on the Xoom. Notably, the presence of measurement variation is detected. If more data points were available from other Xoom devices, configured with a different SW build fingerprints, it would be possible to determine if the software build is the problem or if some other parameter needs to be identified and measured.

Third, the Desire HD has the next highest performance level. While it also has substantial measurement variation, it still produces higher performance than the Droids, LG-P509 or SGH-T959V devices. Of that last group all of them are on the same performance time curve except the LG-P509 which had the lowest performance.

Finally, the GT-P7510 and LG-P509 tended to have the smallest measurement variation. Although there are not enough data points for the LG-P509 to conclude this in general.

4.3 Sort and Sort-Tiny Tests

Two additional CPU intensive tests were created. Both fill an array with random numbers and then sort the array. The array contains data of

type double. The sort-tiny test has an array size of 256 elements and was intended to fit inside a typical L1 cache. The sort (big) has an array size of 10,000 elements and was specifically intended to be well in excess of typical L1 data cache sizes. In this way, the memory subsystem of the processor can be evaluated and perhaps some information about OS interaction with the threads can be observed. Figure 4.5 shows the dual-core GT-P7510 sort-tiny data labeled by thread settings.

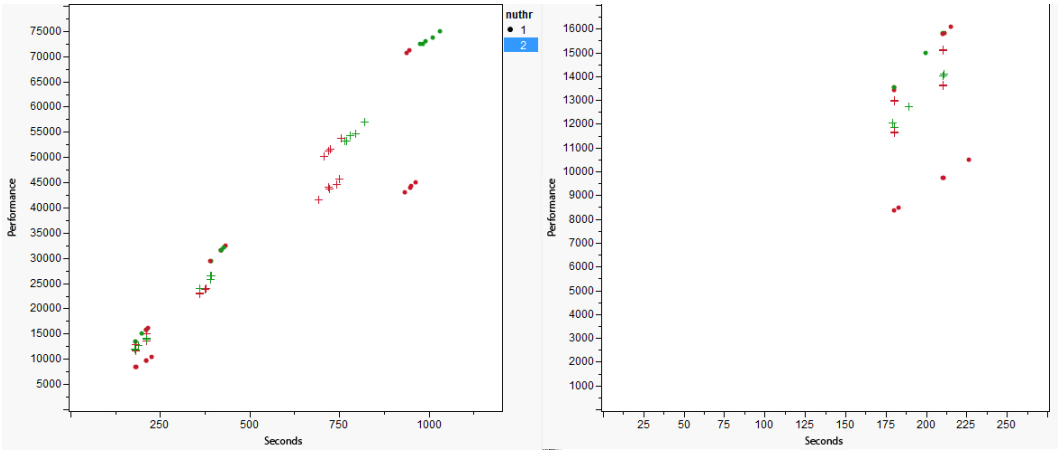

Figure 4.5: GT-P7510: sort-tiny performance vs time for different thread settings

Data points with less than 250 Seconds per 1% battery life had the power management set to on-bright-on (screen on max brightness). Data points with approximately 350-450 Seconds per 1% battery life had the power management set to on-dim-off. Data points greater than approximately 600 Seconds per 1% battery life had the power management set to on-off-off (CPU on, screen off). Note that when the screen is off, the single thread scenar-

ios run for more time than dual thread scenarios. Similar to the observation with the cpu-float test, the trend is consistent with power management mechanisms reducing the power of the second core when it is idle. However, there is a significant difference as compared to the cpu-float test in that running two threads did not yield a performance advantage. In fact, running a single thread with low thread priority yielded the highest performance level per 1% of battery life. A high thread priority tends to give lower performance. The behavior of the OS and the overhead associated with task switching, due to the larger memory footprint, had a notable impact on performance.

Figure 4.6 shows the data for the cpu-sort test. Interestingly, the performance difference between different thread priority settings is greatly diminished. In particular, for single thread cases, the performance and time per one percent battery life consumed is nearly independent of thread priority. Although, closer inspection suggests that, when the screen is on, the variation tended to be lower for the lower thread priority runs and, when averaging is included, slightly higher performance was achieved.

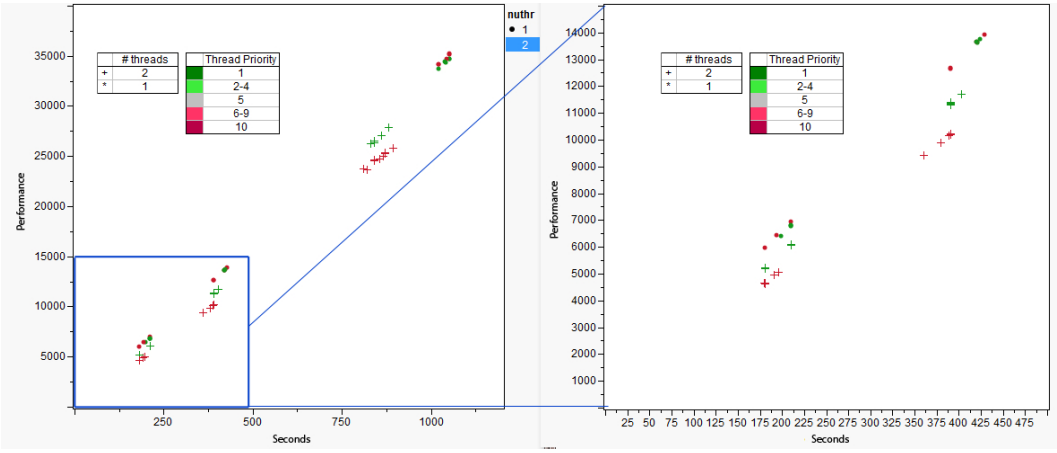

Figure 4.6: GT-P7510: sort performance vs time for different thread settings

Running two threads reduces the amount of time corresponding to 1% battery life. This implies that power has been saved by turning one core off (during single thread scenarios) when it is not in use. This is consistent with the cpu-float test observations in section 4.2. The resulting loss in time yields correspondingly lower performance. The data points for all thread priority and number of thread settings are on a similar performance time slope. This is very

different from the cpu-float test observations of section 4.2. This, combined with the observation that the difference in performance was reduced for the larger array size (sort vs sort-tiny), indicates that the memory subsystem performance is dominating the benchmark's performance. The data points collected on the GT-P7510 tend to be tightly grouped, implying low variation.

Figure 4.7 shows the sort-tiny test data for all devices as a function of the power management setting used.

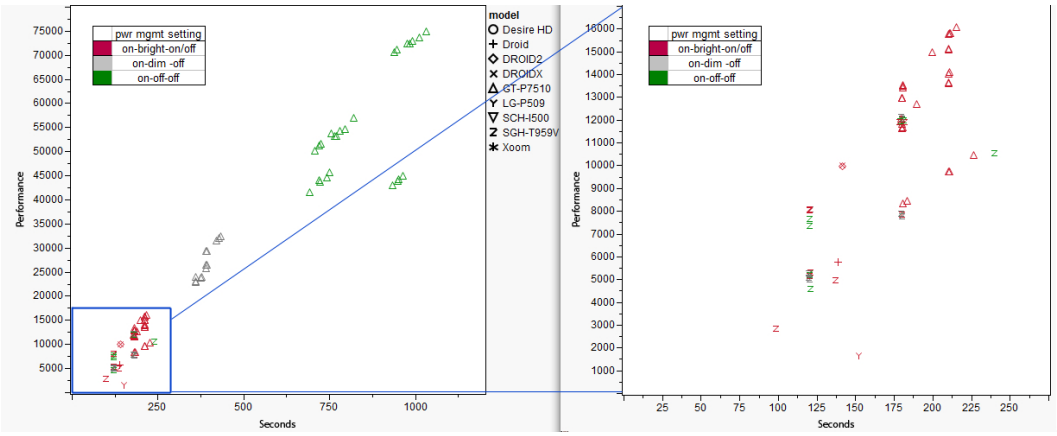

Figure 4.7: All Devices: sort-tiny performance vs time for different power management settings

The GT-P7510 has the highest performance capability. The number of data points for other devices is quite limited; more data is required to draw any reasonable conclusions. However, the data collected tends to indicate that the Droid2 and DroidX are performing well. The data points are on top of each other and at a time of approximately 140 Seconds and a performance level of about 10,000. The Droid data point is about half the performance of

the Droid2 and DroidX. The LG-P509 is the lowest performing device.

The SGH-T959V has quite a few data points and Figure 4.8 shows these data points as a function of the number of threads run simultaneously. Notably, this is a single core device.

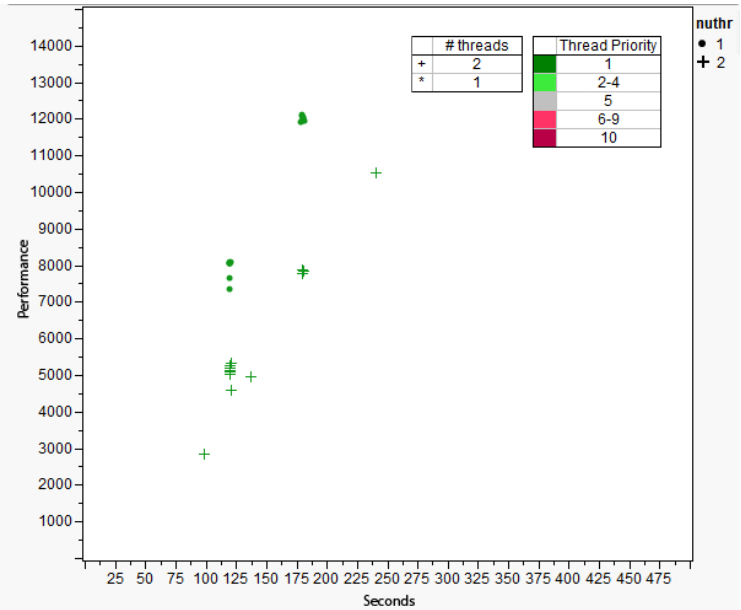

Figure 4.8: SGH-T959V: sort-tiny performance vs time for different thread settings showing bimodal variation in the measurements

Interestingly, all 15 of the single thread data points are from three data records - showing a very bimodal variation in the measurement. Similarly, all 15 of the dual thread data points are from three data records showing another relatively bimodal variation in measurement (with two additional outliers).

Figure 4.9 shows the results of merging (averaging) the data points contained within each of the data records.

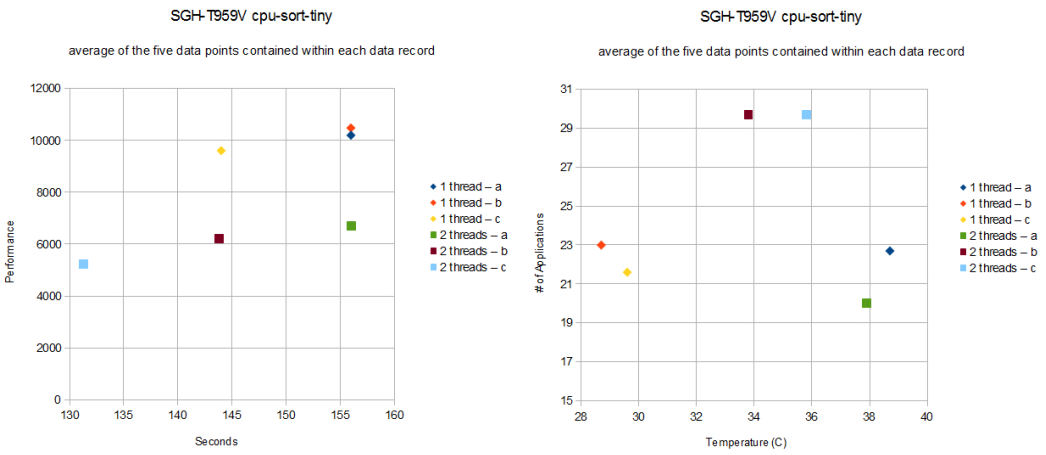

Figure 4.9: SGH-T959V: sort-tiny averaged data points within each data record and correlation to temperature and number of active applications

All data points appear to have correlations to temperature and the number of active applications that is expected based on the discussions in Appendices A and C, except on data point *1 thread-c*. Specifically, *2 thread-b* and *2 thread-c* have identical quantity of active applications, but *2 thread-b* is lower temperature and ran for more time on 1% of battery life. While *2 thread-a* is at a higher temperature than the other (2 thread) data points, it also has $\frac{2}{3}$ the number of active applications. No differences in other measurements help explain the one data point which goes against the expected correlation trends. Finally, it is pretty clear from the data that the two thread scenarios for this device yield lower performance than the single thread case.

Figure 4.10 shows the sort (big array) data for all devices as a function of the power management settings.

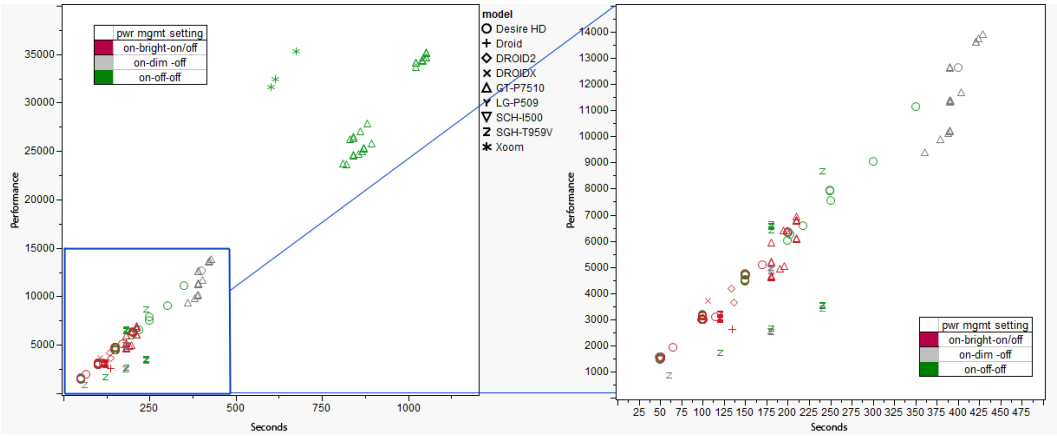

Figure 4.10: All Devices: sort performance vs time for different power management settings

Notably the Xoom's data points stand out as having very high performance and the dual thread cases for the SGH-T959V are the lowest performing. The Xoom had three fewer active processes than the GT-P7510. The Xoom also had a slightly lower percentage of CPU utilization, but both systems were close to 50% (for single thread cases). It would be interesting to collect more data on the Xoom and try to understand why its performance is so much better than the GT-P7510 as both platforms are based on the same processor SoC (Nvidia Tegra 2). The SGH-T959V appears, at first glance, to have low performance. However, it also has some high performance data points – indicating that the bimodal variation in measurement is probably contributing. Additional data collection, and statistical analysis, is needed to answer these

questions.

In summary, the sort and sort-tiny tests did expose significant performance issues in the memory sub systems of the devices tested. Attempting to run two threads discharges the battery faster and does not produce a performance benefit. Increasing the array size compressed the performance difference resulting from different thread priorities. An undesirable power and performance management behavior, occurring when a high thread priority is used, was also observed. The request of a high thread priority results in lower performance for both single and dual thread scenarios. It was shown that the additional parameters of temperature, number of active applications and wireless network connection may be used to help explain differences between individual data measurements, even though they're not directly controlled. There may be some additional factors that need to be identified and measured to improve correlation of the results. Additional data points from more devices would aid the correlation work. Nevertheless, this is a highly encouraging result showing that a great deal of insight about the devices may be obtained from the MobilePowerBench application.

4.4 Pandora Test

An internet streaming benchmark was included where the Pandora application is started and then the MobilePowerBench application is run. To ensure data quality this test checks the list of active processes at each battery level update to ensure the Pandora process is truly active. If Pandora is not

active, then the run is canceled automatically and the last data point disregarded. Data for two devices was obtained and is shown in Figure 4.11 as a function of the power management setting.

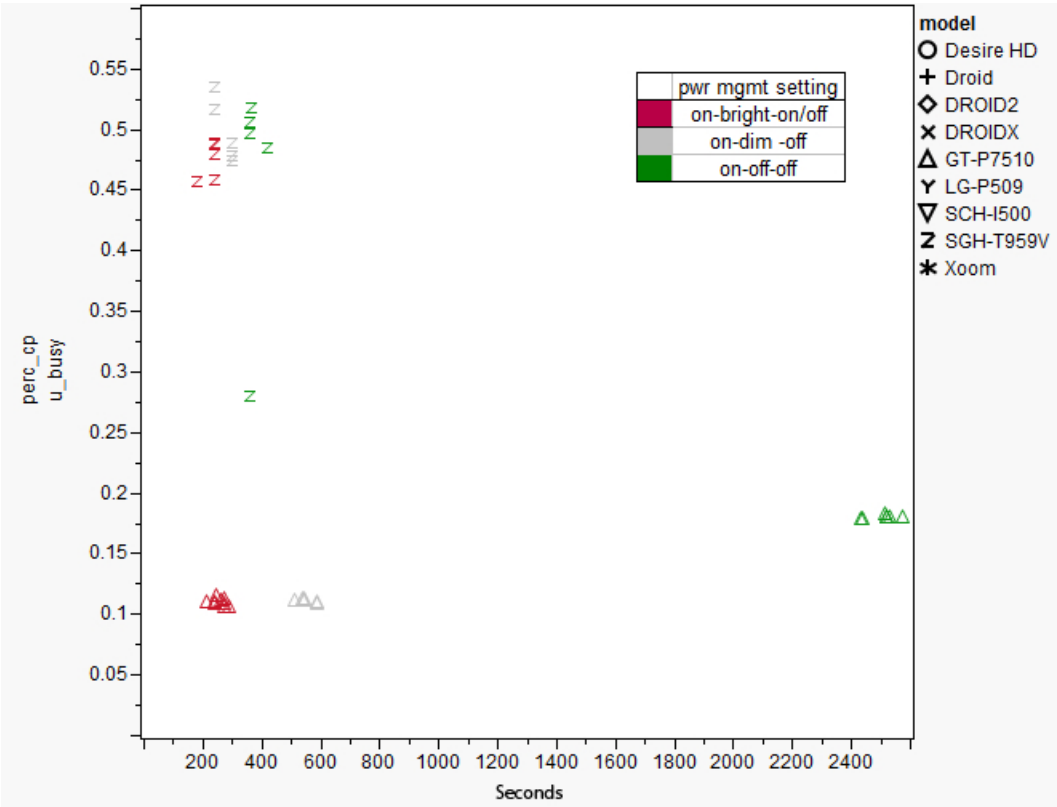

Figure 4.11: All Devices: Pandora CPU utilization vs time for different power management settings

Performance for this test is not applicable assuming the audio quality is reasonable. Thus, the number of Seconds per 1% battery life is plotted against the normalized CPU utilization. The SGH-T959V exhibits little difference in power consumption between the screen on max brightness, dim and off. Also,

CPU utilization is quite high at 50%. This almost implies that either there were other user applications running in parallel or the CPU was doing most of the audio work without a dedicated audio decoder in hardware.

In contrast, the GT-P7510 shows just 10% CPU utilization when the screen is on, battery life more than doubling between a max bright and dim screen setting. However, when the screen is shut off, the time for 1% battery life increases by more than 4x compared to the dim screen. The amount of time the Pandora is able to run on 1% of the battery life is about 2x longer than the longest time for all of the CPU tests discussed earlier as illustrated by comparing with Figure 4.1. The CPU was still held on by the power management setting, so it implies that the power management system is shutting down at least the active power portion of both processors for periods of time. It's also notable that the percentage of CPU utilization goes up, when the screen goes off. While it might be assumed that this is due to additional OS overhead, the CPU utilization data indicates that the increase is in CPU time allocated to user applications. Finally, the measurement variation is quite low indicating that the software conflict discussed in section 3.6 are not a significant problem at lower CPU utilization levels.

4.5 3D Pyramid Test

An OpenGL test was created for a simple 3D object (pyramid). This is continuously rendered as its position and angle changes giving the appearance that it is tumbling across the screen and bouncing off the four edges of the

screen. Color gradients are applied to each surface in a unique way so all surfaces are different as shown in Figure 2.5. The performance metric is the number of frames rendered vs the amount of time, for each percent of battery life, consumed. Figure 4.12 shows the number of frames rendered vs time.

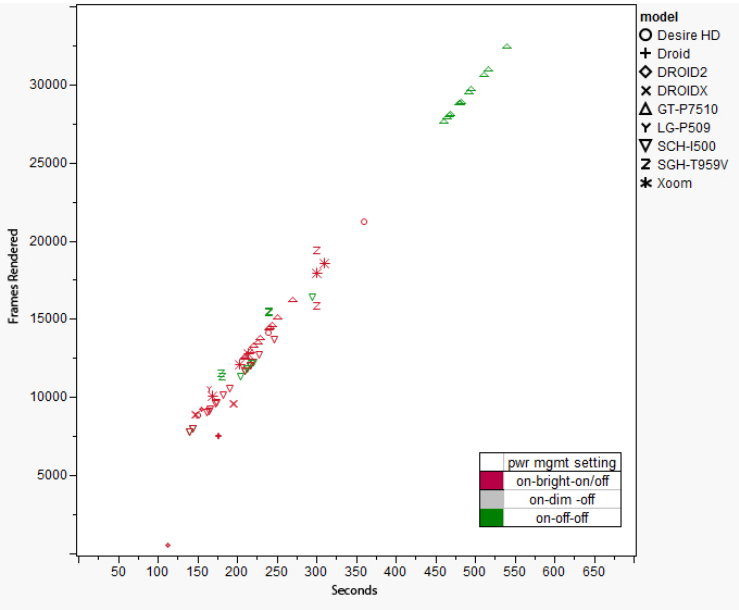

Figure 4.12: All Devices: 3D Pyramid Frames Rendered vs time for different power management settings

Most devices tested are all close to being on the same line - yielding roughly the same FPS rating. This fundamentally means that a graphics test that requires significantly higher performance is needed to properly benchmark the graphics performance of the various devices. Alternatively, it could mean that there is some implicit factor in the rendering code that's limiting the frame rate to a relatively specific value. Additional effort, for this class

of benchmarks, is required to gain more insight regarding the performance and power differences between the devices while running graphics intensive scenarios.

4.6 Temperature and Wireless Impact

There is an interesting set of data points from the GT-P7510 where three data records having identical input settings also have substantially different results. The input settings used were: number of threads set to one, thread priority set to one, test set to cpu-float, and power management set to on-off-off. All data points from the three data records have nearly identical CPU utilization values ranging from 50.02% to 50.08% across all 15 data points. Figure 4.13 shows the data.

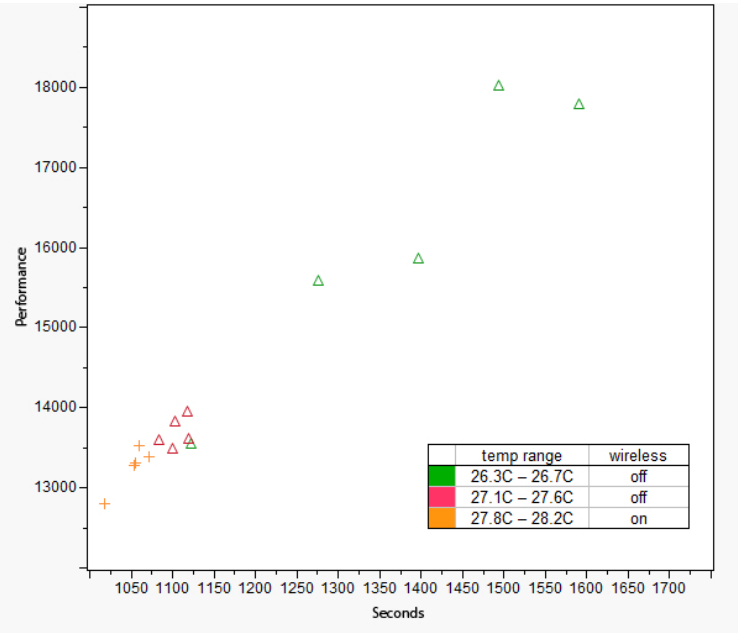

Figure 4.13: GT-P7510 cpu-float test with identical user controlled inputs, showing correlation to temperature and wireless parameters

As expected, with the wireless communications on and a higher operating temperature, the time until 1% battery life is consumed is the shortest. A shorter battery life time results in lower performance because there's less time for the benchmark to run. However there is a significant difference between the red and green groups of data - the only correlation parameter being temperature. Lower temperature should result in lower leakage and thus longer battery life (Appendix A). It is unexpected that approximately one degree C would results in a 30% increase in processing time to consume 1% battery life. It is possible that there was a larger temperature gradient inside the platform and that the temperature difference at the Silicon was actually much lower

than the temperature indicated by the battery. It is also observed that the variation increased substantially and it's likely that some other factor (not being monitored by MobilePowerBench) is contributing to the change.

Chapter 5

Conclusion

5.1 Summary of Contributions

This project has completed the feasibility stage - proving that the business and technical challenges can be solved while achieving the main objectives.

- it's possible for a device to benchmark itself through a stand-alone application with enough resolution to identify power and performance differences between devices
  - data from virtually every Android device in use can be obtained
  - power and performance is measured for different usage scenarios
  - expensive analytic hardware is avoided
  - users without a technical background can contribute results
  - data is anonymously uploaded to a central database
- it's possible to obtain enough measurement resolution to observe the effects of variation and the power management system
  - device variation due to temperature, wireless activity and CPU work-load has been observed and correlated with measurements

- screen settings (bright, dim, off) dominate battery life
- a requirement for strict control over all variables is avoided
- measurement variation can be detected and comprehended through the use of multiple data points
- thread settings and different tests generate data capable of providing power management behavior insight

5.2 Discussion

This report introduced a novel approach for benchmarking Android based mobile hand-held devices. The MobilePowerBench application is capable of running on many different devices. Data from nine unique model types was generated by 20 people during beta testing - two of these devices were tablets. Three different Android API levels (8, 10, 12) were represented. Prior to beta testing, functionality was demonstrated on an Android version 1.6 (API level 4) device; but data using the final code revision was not available for inclusion. 658 data points from 168 data records were uploaded to the central database by beta testers and used for the data analysis in Chapter 4.

Beta testing was completed by the author and the author's friends and family. Several of these users have a technical background, however, a few people without a technical background were able to install the application from an email and run it. This demonstrates that the application is accessible to nontechnical end users. All data uploaded to the Google Appengine was

completely anonymously. The individual user's results are displayed on their device so that they may observe if the performance changes over time.

Some devices, the Droid, Droid2 and DroidX, were observed to update the battery level status in 10% increments. The increased run time on these devices appears to discourage people from generating a lot of data. Only 16 of the 658 data points were from Droids and each of those data points were obtained from unique data records. As a result it's very difficult to determine whether or not measurement variation was a factor for these devices. The MobilePowerBench application greatly reduces the run time from traditional approaches that drain the battery from 100% to 0%. However the run time is still high enough for these 10% devices that it may be a barrier in production - limiting the amount of data obtained from devices with this issue.

The application is capable of measuring both power and performance during many different usage scenarios ranging from CPU utilization of 10% to 100%. The technical obstacles associated with using the device to both test and measure itself were overcome through the use of multiple data points. Measurement variation is easily detected when more than one data point is included in a single data record. In the presence of measurement variation, the mean of data points within a data record may be used to facilitate comparisons between the measurements (Figure 4.9).

Given a properly constructed set of tests, it is possible to observe the performance and power differences between devices - at the system level and Silicon component level. This is accomplished without any analytic lab equip-

ment. A few specific examples are:

- Pandora battery life (Figure 4.11)
  - screen off: GT-P7510 is 2,500 Seconds, SGH-T959V is 400 Seconds
  - screen bright: both GT-P7510 and SGH-T959V are 250 Seconds
- cpu-float test has about five different performance power curves (Figure 4.4)
  - SCH-I500 achieves the highest performance in the shortest time when not heavily loaded by other user processes; however when heavily loaded the performance falls to the bottom tier.
  - GT-P7510 with two threads achieves half the performance as the lightly loaded SCH-I500 scenario in the same amount of time with a bright screen. However, turning the screen off facilitates the highest overall performance on the GT-P7510. Note that data points are missing for the Xoom with dual threads and the SCH-I500 with the screen off.
  - GT-P7510 or Xoom with one thread is on a slope about 60% lower than the GT-P7510 with two threads. But the extra battery life time achieved because one core has been power managed is converted into extra performance. Still, for this test, two threads beat the performance of one thread and in less time.

  - The Desire HD achieves less than half the performance of the GT-P7510 when both have the screen on bright. However, when the screens are turned off, the performance is about 20% of the dual core GT-7510 per 1% battery life.
  - All the other devices tested fall in line at approximately half the performance of the Desire HD.

- Turning the wireless off, for the GT-P7510 running the cpu-float test with the screen off, extended the battery life by approximately 10% as shown in Figure 4.13.

It was also shown that several key parameters such as temperature, wireless activity and the number of active applications, while not explicitly controlled, may be measured and provide useful correlation information (Section 4.6). This helps explain the differences between measurements having identical user controlled input settings.

With additional test writing it may be possible to use the MobilePowerBench application as a tool to evaluate software and hardware power management interactions - leading to improved system performance and the ability to monitor the changes in the field. It should also be possible to superimpose an individual device's data over all of the anonymous data uploaded by other users so each person can see how their device compares with other devices. This would enable an end user to compare their particular device

against the statistical distribution of data produced by other end users with the same model.

Additional work is required to take the project from prototype to production and there are no business or technical issues preventing this. The fundamental objectives of the project have been achieved and proof of concept demonstrated.

5.3 Future Development Required for Production

A few capabilities need to be added to the MobilePowerBench application before a production release to the general public.

First, the creation of a leader's board - some way for users to see how their particular device compares to other devices. This would help keep people's interest in uploading their data. There are several comparisons that users are interested in: (1) comparisons of several devices from the same manufacturer and the same model number, (2) comparisons of all devices from the same genre (*i.e.*, smart-phone or tablet) or (3) comparisons of all devices. Furthermore, if a device has poor performance or battery life, it is desirable to offer possible explanations based on correlation of the other measured parameters (such as temperature, number of processes, etc.)

Second, a method of securing the data in the Google App Engine is required. Because the data may be uploaded without a login, to maintain the anonymity of the users, it also means that hackers and/or unscrupulous

manufacturers might be tempted to upload large quantities of bogus data to skew the results. Thus, securing the data is required prior to a production release.

Third, additional benchmarks are needed to cover other common and interesting usage scenarios. Examples include: (1) Skype, (2) web-browsing, (3) watching music videos, (4) watching digital video stored on the device, (5) playing games, (6) additional graphics intensive tests.

Finally, incremental improvements to the controls and extra parameters being monitored should be explored. For example, it makes sense to prevent the user from launching more threads than there are CPU cores in their particular device. There may be other parameters that should be monitored to help improve the explanations of variations observed in the results. For example, interrupt handlers and power consumption are disrupted if the device spends a great deal of time searching for a wireless connection.

Appendices

Appendix A

Technical Issues Leading to Power Management

In order to fully grasp the power management problems, three basic areas must be understood:

- Transistor Characteristics, Constraints and Impact
  - maximum voltage and temperature (transistor reliability)
  - minimum temperature (races, reduced performance, system costs)
  - minimum operating voltage (state-retention)
- Power Consumption
  - dependence on transistors, voltage and temperature, and in-die variation (IDV)
  - pre-silicon design predictions and modeling
  - measurements during post-silicon debug and production testing
- Power Delivery
  - LRC network from the voltage regulator module (VRM), through transistors, and back (ground path)

- 1st, 2nd, 3rd droop - corresponding to die, package, main-board parameters
- voltage drop due to static currents (IR drop) vs drop due to transient currents (di/dt droop) which narrows the window between min and max voltage for operation.

The interactions and trade-off options between these issues of the design are highly complex. Missing the power targets or failing to handle the power-delivery issues properly during design will influence the final performance experienced by the end user. Reduced battery life and higher electric bills may or may not prevent the product from going to market; nevertheless, the processor may not be competitive enough if the frequency must be reduced in order to meet the thermal or power delivery constraints of the overall system. Striking the right balance between the various costs (design, manufacturing, test, system) is another key element of the design process. How much will customers pay for the extra performance that comes from a processor with a more robust power management system? How much time will it take to add certain power management features in to the design and will the market window close before the silicon is ready?

A.1 Transistor Characteristics, Constraints and Impact

Every process technology has voltage and temperature sensitivities. The performance and power of the processor will vary significantly with changes

in voltage and temperature Furthermore there are limits (maximum and minimum) for conditions of voltage and temperature which the processor must stay within in order to function properly (without generating errors).

A.1.1 High Voltage and Temperature

Typically the maximum voltage and maximum temperature constraints are driven by device reliability constraints. State of the art foundries are developing processors using transistors with minimum feature sizes of 32nm to 20nm today. Typically this directly refers to the minimum drawn channel length of the gate (from source to drain), of the MOSFETs. Often transistors in a technology node may be offered with slight variations of channel length or threshold voltage to offer design teams the ability to obtain higher frequencies, albeit for a leakage penalty. Several wear out mechanisms which result in processor performance degradation over time exits. Some examples include: hot-e, electro-migration, punch-through, oxide break-down, Bias-Temperature Instability, etc. These wear-out mechanisms are accelerated by high temperature and/or high voltage and foundries utilize Burn-In (exercising material at elevated voltage and temperature) to evaluate the process technology's degradation characteristics[51]. Maximum transistor temperature (at the junctions) is typically set in the 90C - 125C range. Max voltages for transistors with drawn poly gate lengths in the 30nm range are limited to around 1.1V[59]. Minimum transistor channel length devices are typically offered with different VT options to enable design teams to optimize the power and performance[58]. In

order for process technologies to support higher voltages, it must also provide transistors with thicker gate oxides and longer channel lengths[13]. However these devices are much slower and consume more area than the minimum device size transistors. Nevertheless, if the processor or System-on-Chip (SoC) desires to communicate using certain standard protocols (like USB) they must design IO interfaces that use transistors capable of long term reliability at higher voltage levels. Frequently these devices used for IO are optimized for analog circuit performance instead of maximum switching frequency like the minimum feature size transistors are for digital circuits.

A.1.2 Low Temperature

For years, processor frequencies have been improved by reducing the temperature[65]. Over-clockers have exploited this attribute and liquid cooling systems are commercially available for enthusiasts to add onto their systems. Other companies such as the Alienware desktop system by Dell Corporation actually build high-end systems that come over-clocked with liquid cooling. There is research toward the development of micro channels which allow cooling fluids to flow through the Silicon chip, although this work is focused more on compensating for increased power densities resulting from process scaling[12]. These systems may be run at high voltage for long periods of heavy utilization without exceeding the temperature limits of the transistors.

However, device scaling over the years has pushed maximum voltages, allowed by the reliability limits of the smaller transistors, down relative to the

transistor's threshold voltages. Products are experiencing reduced frequencies during low temperature conditions, particularly when operating at lower voltage levels. This can be understood by exploring the temperature dependent terms of the drain current equation for a MOSFET:

$$I_{DSAT} = \mu\ C_{ox}\frac{W}{2L}(V_{GS} - V_T)^2 \tag{A.1}$$

Mobility (μ) goes down with higher temperature - more energy in the crystal lattice results in a shorter mean-free path for free carriers moving through. However transistor V_T decreases with higher temperature and thus the $(V_{GS} - V_T)^2$ term in the equation A.1 gets larger as temperature increases. Historically V_{GS} (equal to the supply voltage when the digital transistor is switching on) has been quite large relative to the values of V_T such that the changes in V_T had a much smaller impact on the drive current of the transistors. Now it's not uncommon for a product to have the traditional temperature dependency (slower at hot temperature) above some voltage and then the inverse dependency (slower at cold temperature) below that same cross-over voltage. This also means that transistors with different V_T values (HV_T, LV_T, etc.) will have different frequency vs. temperature cross-over voltages. Interestingly, certain customers require extreme cold and/or hot operation limits for example: military, automotive and wireless/cellular. These customer driven design parameters can have a substantial impact on both circuit and architectural decisions made during design. However, this fundamental shift in the temperature dependency now means that performance may actually be lost if

the temperature drops and this must be comprehended by power management schemes today.

A.1.3 Low Voltage

Typically the minimum operation voltage is determined by the ability of the state elements in the design (memory cells, flip-flops, latches, etc.) to retain their value over time. There are at least two interesting minimum voltage metrics related to memory elements - retention and active minimum voltage. Retention is the minimum voltage required to maintain a steady state when the cell is not being written to or read from. Active is the minimum voltage required to ensure the cell may be written to or read from with some minimum acceptable timing (usually corresponding to the minimum frequency obtained by the processor at the minimum active voltage point). The retention voltage is usually lower than active voltage, but they both depend heavily on the circuits and process technology used. Specifically the minimum voltages depend on the variation (random and systematic) in the process technology, the number of bits in the design, and the specific circuit topologies employed. To understand the variation impact, suppose a memory cell's minimum voltage of operation is 0.7V at the target temperature condition and the typical process corner. After variation is included in the memory cell's performance analysis, there might be a one in a million chance that the minimum voltage of any one memory cell is actually 0.75V. If there are 500,000 memory elements in the design (about 64KB) then the yield to a minimum voltage target of 0.75V

would only be about 50%. Notably, this is much worse than the desired 0.70V minimum voltage. The processor's minimum voltage can be modified by:

- Changing the transistor topology of the memory cell (interruptable bit-cell structures can generally be written at lower voltages than jam bit-cell structures)
- Changing the sizes of the transistors[54]
- Reducing the V_T of the transistors used in the array, read and write logic. This generally has an unacceptable impact on overall processor leakage (due to the strong sensitivity of leakage to V_T)
- Adding error correction circuitry can be used to help achieve lower voltage - usually with a penalty in power, area and performance (latency to do the correction in-line)

Reducing the minimum voltage required for operation is an active area of research[41]. Other projects are actively making circuit design changes to help achieve lower minimum voltage. One high performance processor was able to scale down to 0.7V using 8T SRAM cells, instead of 6T[63].

Thus there are upper and lower limits to voltages and temperatures that may be employed by a design team. Processor frequency is dependent on both temperature and voltage in a non-linear way. Often process technologies are supporting multiple transistor variants. Each variant will have different temperature, voltage constraints and frequency interactions. Not only does

this apply to fundamentally different transistor architectures, on significantly different pitch, but it also applies to digital transistors implemented with the minimum pitch but variants using either different V_T or L.

A.2 Power Consumption

A comprehensive treatment of the various ways CMOS circuits consume power is well documented elsewhere[60]; however a brief discussion of the key components is included as some of the terms and concepts are relevant for the power management treatment later.

A.2.1 Leakage Power Components and Dependencies

Leakage power of MOSFETs have contributions from several different mechanisms of which just three will be discussed here: subthreshold, gate and junction leakage.

Subthreshold leakage occurs between the drain and source when the gate is off $V_{GS} = 0$. This usually dominates the other leakage mechanisms in terms of magnitude. It is a strong function of transistor V_T and L - which can have significant variations in modern process technologies. Subthreshold leakage power has a strong dependance on V_{GS} and V_{DS} during operation. Because V_T has a strong dependence on temperature, subthreshold leakage is also highly dependent on temperature (lower temperature implies higher V_T and thus lower subthreshold leakage).

Gate leakage occurs between the gate and other three terminals of the

MOSFET (the drain, the bulk and the source). This is a strong function of the gate-oxide material, thickness and quality. Often gate leakage is much lower in magnitude than subthreshold leakage. However, gate leakage may be relevant if the voltage is very high and subthreshold leakage is low - as may be the case if devices with high V_T or long L have been used. Also, low temperature operation may reduce subthreshold leakage enough that gate leakage must be included in power estimation calculations.

Junction leakage occurs between the bulk and drain terminals as well as between the bulk and source terminals of the MOSFET. This is due to the pn junction formed between the well and the source/drain implants. It also depends on the voltage across the pn junction. Many of the transistors in the design will have a zero (or near zero) voltage across the source to bulk pn junction as these are often electrically connected. Some transistors in the design will have non zero voltage across the drain to bulk pn junction. Generally, junction leakage and gate leakage are much lower in magnitude than subthreshold leakage unless very high V_T devices and/or body bias techniques are being used.

A.2.2 Active Power Components and Dependencies

Active power of CMOS circuits depends primarily on the relationship shown in equation A.2.

$$P_{active} = CV^2F(AF) \tag{A.2}$$

The components of this equation are defined as follows:

- C: the amount of capacitance switching
- V: the voltage change that the capacitance experiences
- F: the frequency of the highest frequency clock signal in the design
- AF: the Activity Factor, used as a de-rating for signals that do not switch as often as clock signals.
  - a clock signal has $AF = 1.0$ (two transitions every cycle)
  - a data signal that changes once every cycle has $AF = 0.5$
  - a signal that never changes state has $AF = 0$

There are other factors contributing to active power consumption that may not be neglected: short-circuit (or rush-through) current, glitch, bias-currents (in analog circuits), and active leakage.

Short-circuit (or rush-through) power is wasted energy flowing straight from the voltage supply to ground during a switching event when both the PMOS and NMOS devices are on ($V_{GS} > V_T$). This is dependent on the V_T of the transistors, the operating voltage, and the slew rates associated with the inputs and outputs of the gates.

Glitch power occurs when inputs to a gate arrive at different times within a clock cycle, or when there is a noise event. A circuit node may switch multiple times in a given clock cycle when a glitch occurs. In the worst case, glitch power may result in full voltage transitions on certain nets in the

design. The additional power will be of the magnitude given by equation A.2 associated with the capacitance (C) of the nets experiencing the glitches. In minimal impact cases, the glitch may be small such that the transistors do not fully turn on ($V_{GS} < V_T$) and the magnitude of the power is a multiple of the leakage currents associated with the devices experiencing the glitches.

Analog circuits use current bias circuits for their amplifiers. These bias currents may represent a significant portion of the power associated with the circuits. Furthermore, these are static currents that consume power independent of the data transfer rates. The only way to eliminate the DC bias currents is to shut down the circuits; however, there may be significant overhead and performance issues associated with restarting the circuits.

Active power consumption is generally not uniform across the entire silicon die - leading to hot-spots. Because leakage has such a strong dependance on temperature, the transistors in the regions of these hot-spots have higher leakage than the transistors in cooler locations of the die. If there is no active power, then the temperature across the silicon will stabilize to a uniform value and the leakage will change. This may be modeled as either a component of active power or a Non Uniformity Factor (NUF) may be applied to the leakage. It is important to remember that these leakage increases (due to hot-spots) are the direct result of active power. Leakage is not static, it depends on the work being done by the device.

A.2.3 Manufacturing Variation Effects

Manufacturing variation effects have a pronounced impact on leakage, the voltage required to achieve a particular frequency, and even active power. There is significant manufacturing variation in a single die (IDV), between different die from the same wafer, and between die from different wafers. Channel length (L) is directly dependent on the critical dimensions associated with patterning. This is challenging modern process technologies that are attempting to image 32nm (or even 20nm) features with 193nm light sources. Very small variations in focus, wafer tilt, resist thickness, etc., result in different actual channel lengths or wire widths. Furthermore densities in different regions of the die will lead to different thicknesses of wire after Chemical Mechanical Polish (CMP) steps. Oxide thicknesses are just a few atomic layers thick and the volumes associated with the channel, source and drain regions create a situation where atomic variations of implants and thicknesses have first order impacts on transistor V_T.

The fact that subthreshold leakage has such a strong dependence on these manufacturing parameters, also prone to significant variation, creates an unfortunate situation. The frequency of the entire die is limited by the particular transistors that resulted in the slowest paths of the design. However, the faster transistors on that same die have higher leakage; moreover a 1σ fast transistor adds more leakage than a 1σ slow transistor saves. Thus, the actual silicon die has a lower frequency and higher leakage than would be predicted based on the characteristics of the mean of all transistors on the same die.

Adaptive body bias and Vcc tuning has been shown to help reduce the impact of IDV - recovering yield that would otherwise be lost[30] [26]. This must be accounted for early in the design cycle or the chip will consume more power than predicted. In power constrained designs power of individual components is reduced by reducing the performance - thus excess power consumption may result in a performance loss.

Active power sensitivities to process variations are present but they do tend to be less pronounced than the impact variation has on leakage and frequency through the transistor's channel length and V_T. Capacitance variations are introduced from width, space and thickness variation. This impacts both wires and transistors. In fact transistor capacitance has strong dependencies on the particular process technology architecture decisions made and it will vary with the tight critical dimensions during fabrication[61]. Generally die/wafers with higher capacitance lead to higher power and lower frequencies. It's possible that a die/wafer with lower capacitance may still be slower, for example, if the reduction in capacitance occurred because of a large increase in resistance associated with very narrow or very thin wires.

A.3 Power Delivery

Processor power delivery is fundamentally a LRC network from Voltage Regulator (VR) to the transistors and back again to the VR. Often, VRs are designed and built on a separate component and thus the LRC network goes from the VR, through the mother-board, through the package, through the on-

die metal stack to the transistors. There is a similar return path for ground, through the on-die metal stack, through the package, through the motherboard to the VR. As a side note there has been some work exploring VR for 3D or on-die integration[28]. The current draw of the processor can change much more quickly than the time constants associated with this power delivery network. It is important to study the transient response of the network as the frequency will be limited by it[38], thus a qualitative description of the issues is presented below. Very complex power delivery systems are used in multi-core processors and SoCs today[10].

A.3.1 Steady State Conditions

Under steady state conditions (constant current load from the target device) the inductance and capacitance do not play a role (the current through the inductors is constant and thus the voltage drop across them is zero) and the voltages at the capacitance are constant also. The series resistance does cause a voltage drop ($V = IR$) between the voltage regulator and the transistors. Systems typically use a very low current feedback (from the target device back to the VR) mechanism to enable the VR to adjust the output voltage in order to maintain the desired voltage at the transistors - maximizing performance. Nevertheless, there is a system level power loss associated with this resistance.

A.3.2 Transient Response: First Droop

When the integrated circuit changes it's current consumption a di/dt event is initiated at the transistor level. The inductance and capacitance in the power delivery network (along with the magnitude and shape of the di/dt event) will determine what happens to the effective voltage seen by the transistors. The package inductance prevents an instantaneous current change through it; and thus the on-die capacitance tries to maintain the voltage on the transistors while supplying (or absorbing) the current change. If the transistors suddenly increase their current consumption, the current flowing through the package inductance does not change, and thus the on-die capacitance supplies extra current as the voltage across the transistors starts to drop. This is referred to as first droop, and it's dominated by the magnitude of the di/dt event, the on-die capacitance and the package inductance. Intrinsic on-die capacitance associated with wire cap and diffusion cap exist (and needs to be estimated by the design team). Adding more on-die capacitance consumes area, however there have been proposals to use DRAM like capacitance structures to help mitigate the area penalty[35]. Additional work has been done to determine the spatial influence of on-die decoupling capacitance when the on-die power-delivery network (LR) is accounted for[37].

The voltage droop caused from sudden increases of current consumption (di/dt) also cause the transistors to operate at a lower frequency during this time. Thus, the processor's frequency is limited not by the voltage of the transistors during steady-state, but by the steady-state voltage minus the

worst-case transient droop that can occur. Sadly, this means that the processor must be run in steady state with voltage at the transistor higher than what should be required - so that when the droop events occur the voltage stays at or above the required voltage to maintain the target frequency, otherwise failures will occur. In the best case these failures result in a system crash; in the worst case they result in data corruption that is not detected. Similarly, if there is a di/dt event for which there is a sudden decrease in current consumed by the transistors, the package inductance will continue to feed current into the die and the on-die capacitance will begin to charge. The voltage at the transistors will increase resulting in a transient voltage over-shoot. If the transistors were already operating at the maximum voltage allowed (for reliability reasons) then the over-shoot would accelerate the degradation of the transistors. The maximum voltage allowed for normal operation must be far enough below the maximum allowed by the process so that these over-shoots do not wear-out the device prematurely. Thus, transient current events lead to a reduction in the available voltage range (lower maximum and higher minimum) that the transistors may effectively use.

A.3.3 Transient Response: Second Droop

Some time after a transient current event, the package inductance starts conducting additional current to restore the voltage on die to the target level. The capacitance on the package or on the mother-board at the edge of the package, and the inductance in the mother-board's traces from the package to

the VR have a similar interaction as described in section A.3.2. Depending on the system, these package decoupling capacitance may be significant[55]. Initially, the mother-board inductance current will remain constant and the package capacitance will try to maintain the voltage by either charging or discharging current to make up for the difference created by the change in current going through the package inductor. This second tier response will occur after the 1st response described in section A.3.2 because of the package inductance's effect. If the di/dt event increased the current flow then a second droop in voltage results. Designed properly, the second tier response should have lower magnitude than the first tier response and thus the first tier response limits the available voltage range for the transistors.

A.3.4 Transient Response: Third Droop

Still more time after the initial transient current event, the mother-board inductance will start to conduct and the capacitance in the mother-board trace at the VR output will have an impact. Also the VR will start to adjust the output current corresponding to the change in demand. After some time, the power-delivery system will stabilize as long as there's not another change in current consumption. The magnitude of the third tier response should again be lower than either the first or second droop. So the voltage first tier response still limits the available voltage range for the transistors.

A.3.5 Transient Response: Design Considerations

For modern processors the inductance values are on the order of 0.1nH to 10nH. Package and mother-board capacitance are often in the 10uF to 10mF ranges. This is a wide range but small hand-held devices often have lower capacitance and higher inductance values. Large processors in servers may have large capacitance and low inductance. This is instructive because the system's power-delivery response time is often much larger than the frequency of operation for the processor itself. There could be several di/dt events that occur before the full power delivery system can respond and return to steady-state.

The resonant frequency of the power-delivery system is important. If the processor has some disposition to change power consumption at the resonant frequency of the power-delivery system, it's possible that the system could become unstable. It's also possible that the initial response (dominated by the on-die capacitance and package inductance) is not sufficient to describe the worst case over-shoot or droop events. The result is that more voltage margin (away from max voltage for reliability and away from min voltage required to hit a target frequency) must be allocated by the designers.

One of the key pre-silicon design challenges is to predict the worst case droop and over-shoot events that can occur. Often this is related to the test content discussed earlier and predicting the worst case di/dt event. Because of the high latency associated with a processor exiting very low power idle states, the worst case di/dt events usually are associated with transitions from

a low power but active state to a high power and active state in just a few core clock cycles. How quickly the processor goes from the minimum activity test condition to the power virus test condition is one example. There may be other conditions that are worse, making it important to identify these in advance of the power-delivery system design closure.

It might be expected for this to only be a problem for very high-power systems, running at high frequency, because they have di/dt events with the highest amplitude that can occur in very short periods of time. However, these systems are also larger and less cost sensitive. These systems are able to afford packaging methods that reduce the inductance and increase the capacitance - helping mitigate the problem. Low power processors also suffer from exactly the same effects greatly impacting the overall system. Even low power processors used today operate in the GHz range and require multiple power supplies[1]. While the magnitude of the di/dt events may be 100 times lower than that of a large processor, the small form factor and cost pressure results in much higher inductance and much lower capacitance values in the power delivery system. Furthermore, the voltage margins that must be allocated to compensate for these issues directly impact the product's power consumption (impacting battery life) and frequency (performance observed by the end user). These are critical metrics in today's hand-held devices. The power delivery issues facing design teams are as relevant for small, low power devices, as they are today for high-power, multi-processor systems.

It is important to recall that the manufacturing processes compensate

for frequency and leakage variation by tuning the target voltage or adjusting the body bias of the transistors[30] [26]. This is important because the slower units (with less leakage) are operated at a higher voltage in order to achieve the target frequency. Similarly, units running at cold temperature may be operated at higher voltage to maintain frequency - when the leakage is low. These units have an active power that makes up a larger percentage of the total power. Thus, the worst di/dt events occur for the slow material operating at cold temperature and higher voltage levels. The condition that creates the greatest di/dt event may not be the condition that leads to the largest peak power in steady state.

A.3.6 Additional Design Considerations Required for Power Delivery Analysis

The power delivery system must be capable of meeting the peak power consumption in the end-user system. However, at least portions of it (the die and package) must also be capable of operating under more severe test conditions. The test environment is important because the manufacturing team will need to operate the processor at voltages, temperatures and frequencies well above what the end-user systems will experience. Quality and reliability teams run material at elevated voltages, frequencies and temperatures to accelerate wear-out mechanisms. This allows them to determine if the product will meet the expected lifetime requirements demanded by the customer. If the units, or some sampling of the units, cannot operate at these conditions because of a power-delivery problem then the qualification of the product can not be com-

pleted and it could trigger another spin of the design. If performance monitors or other circuits are enabled in the system, the power delivery system must accommodate those additions. The maximum power used in end-user systems is critical because it sets the fundamental limits on the power supply required to meet the worst case demand.

Voltage regulator efficiency is also very important. Typically a voltage regulator can be tuned to have a peak efficiency of around 80-85% over a certain current/load range. Furthermore, individual supplies on the processor will often require that the VR be capable of delivering a range of possible voltages and currents. Often it is desirable to maximize VR efficiency for the current loads associated with the TDP power associated with realistic applications. The efficiency might drop from a peak of 85% to 50%) across the range of possible current loads. This also implies that it's advantageous to have the VR efficiency either be as wide and broad as possible. Sadly this usually leads to a lower peak efficiency. It is possible to design a VR system whereby the efficiency can be changed depending on the actual load current. This can be done by either switching between different passive components or changing the number of phases used in the VR[48]. While this works well going from high current load to low current loads, it leads to higher voltage droops and a worse power delivery response. Also note that the time associated with changing the VR efficiency point is much larger than the time it takes the processor to generate a di/dt event. There are situations where the processor knows that a change to a different power state is going to happen and the VR

may be notified to change the output voltage and to change the expected peak current supported.

Appendix B

Pre-Silicon Design Prediction and Estimation

Power conscious design requires a method of estimating the power before the system has been completed. Different components will consume a different amount of power depending on the performance level required. For example, a screen on maximum brightness will consume considerably more power than the same screen at a dim or off setting. Another example is the difference in graphics power consumption between playing a game and web-browsing. The game scenario requires high graphic processing and frequent display updates but the web-browsing scenario infrequently updates the display. These real world usage scenarios the system experiences indicate a need to estimate the power indicate a need to estimate both performance and power under these scenarios; however, early in the design cycle, when major architectural decisions are being made, little may be known about the performance and power of the components.

To solve this problem methods of estimating power at each level of abstraction and corresponding accuracy are needed. Evaluating each component in the system by finding the upper, lower, and typical limits of power and performance is useful. This information is useful to both guide the component

design teams toward power reduction and to bound the power and performance limits. The method enables a team to confidently build a design that works with the predicted power and performance. This also allows teams to predict intermediate scenario power. This is an iterative process and intermediate results are used to identify dominant and problematic components. Consequently additional effort is expended by the design teams toward refining the accuracy of the estimates and toward reducing power consumption. Thus, there are both different levels of abstraction required for power modeling and different work load conditions that must be analyzed to bound the power (and performance) limits.

B.1 Work-Load Terms and Corresponding Power

Within any program a processor executes, there will be times of high activity and times of low activity. High activity typically correspond to times where the execution units can work on a continuous stream of instructions and data. This occurs when the processor is operating near its maximum performance levels (maximizing IPC) and it is the result of accurate branch-prediction, instruction and data pre-fetching (all of the data available in the local register files and/or first level cache). Processor architects work very hard to maximize IPC and ensure the processor spends most of its time in this state when there is demand from the user. Low activity occurs at times when the processor stalls or when user demand is low. There are some definitions of terms used to describe various work loads and how those correspond to

processor power. It's important to note that these definitions may either include or exclude leakage power.

B.1.1 Virus Power

The virus power corresponds to the maximum power the processor can consume using instructions available to all end users. More specifically the active power equation A.2 may be thought of as shown in equation B.1.

$$P_{active\text{-}virus} = C_{eff\text{-}virus} V^2 F \tag{B.1}$$

Note that the AF term from equation A.2 is equal to one, more accurately the term has been (effectively) included in the $C_{eff\text{-}virus}$ term of equation B.1.

Generally the virus is a synthetic code stream that is designed specifically to maximize power consumption instead of doing any useful computation. A very tight loop of instructions keeping all the required data in the local register files and keeping as many execution units active as possible. By maximizing the IPC it's possible to maximize the amount of capacitance in the design that switches every cycle. There may also be branches, cache snoop traffic or data movement - but it's done in a way so as to ensure the execution stream is not slowed down. This code must be written by someone on the design team that has a strong understanding of how to maximize the performance of the processor. The person must also understand which units contribute the most amount of power as this information is not available to end users. The test, while not doing anything particularly useful, is sustainable. Thus, it is possible to run the test for an extended period of time, very long relative to the

power-delivery and thermal system response time constants. The power delivery system must be designed to handle this sustained power level unless there is a power management method used to ensure the processor never consumes more power than can be supplied. There should be a few tests that are competing to have the power virus title (the worst sustainable power consuming test bench) - as they're all useful for power reduction and power management analysis.

The virus test may actually change over time as RTL features are added and power reduction work is accomplished. It is important to have multiple virus test candidates per sub block, because it provides insight regarding which combination of sub blocks may result in the highest virus test. This is also important for giving the design owner of that particular block feedback as to how much power their block consumes enabling them to track their power reduction efforts. These virus candidates which stress the sub blocks of the design, also provide useful input to the modeling of the power management system helping determine the upper bound of power consumption.

B.1.2 Thermal Design Power (TDP)

The Thermal Design Power corresponds to the sustained power the system must dissipate in order to ensure that 'all' realistic applications (of interest) do not exceed the cooling capabilities of the system. By definition, this is always lower than the virus power. The active power portion of the highest TDP code stream is defined as an Applications Ratio (AR) de-rating from the

active power portion of the virus power. Thus equation B.1 is modified as shown in equation B.2.

$$P_{active\text{-}TDP} = (AR)C_{eff\text{-}virus}V^2F \tag{B.2}$$

$$C_{eff\text{-}TDP} = (AR)C_{eff\text{-}virus} \tag{B.3}$$

AR values may vary considerably depending on the processor's micro architecture, the quality of the virus test identified by the design team, the application code streams, and compiler optimizations available. While a design team may have just a few candidate virus power tests that are tracked during processor development, there will generally be a few dozen scenarios of real applications that are tracked for TDP estimation. The key point behind this definition of TDP is that TDP is an attempt to model the highest power that a real application will consume over an extended period of time relative to the thermal system response time constants and the power delivery system response time constants. If TDP power consumption is exceeded in the system for too long, then modern processors will initiate a throttling event intended to get the temperature back under the desired limits. The different methods of achieving this outcome are further discussed in chapter C. It is possible for real applications to consume power in excess of TDP, as long as it's for a short period of time and thermal limits are not exceeded. The set of real applications must include all those that are important to the performance modeling team otherwise thermal throttling events will result in lower system performance than originally predicted by the architecture team.

B.1.3 Idle Power

Idle power corresponds to the power consumed by the processor when there is no activity. Modern processors have several possible idle power states which will be discussed in chapter C. It is particularly important, for mobile devices, that power is both minimized and accurately estimated in these states because mobile devices have many real world usage scenarios with extended idle periods. For example, smart phone data sheets routinely list stand-by time.

B.1.4 Minimum Activity (but not Idle) Power

Minimum activity power corresponds to the power consumed when a small number of operations are being executed continuously. This classification of tests is used to establish a lower power consumption bound of the processor when active. This is significant because there are circuits that need to be active when any work is being performed (*i.e.*, PLL, global-clock distribution, etc.) These tests enable designers to identify circuit blocks not being disabled even though they're not required to get the small amount of work completed. As a result power consumption of real applications may be further reduced. These tests also enable a lower power consumption bound to be identified corresponding to the the lowest activity levels. Completing the thought, equations B.4 and B.5 derive from equations B.2 and B.3 respectively.

$$P_{active\text{-}min} = AR_{actmin} C_{eff\text{-}virus} V^2 F \tag{B.4}$$

$$C_{eff\text{-}actmin} = AR_{actmin} C_{eff\text{-}virus} \tag{B.5}$$

B.1.5 Average Power

Average power corresponds to the average power that realistic applications may consume. It results from averaging the power consumption over a long period of time – where the processor may experience smaller time periods of power consumption ranging from idle to TDP power. One example of an average power benchmark is Mobile Mark - used to estimate the battery life of portable processor systems from hand-held devices to laptops. Another example is TPC-E (Energy) used to calculate the power per transaction on multi-processor server systems - critical to projecting the cost of the end-user facility required to achieve a certain performance level. There is potential for an infinite number of real-world usage scenarios. Design teams must reduce the possible usage scenarios to a small subset believed to yield sufficient feedback to the design process during execution.

It's notable that any average power scenario estimate may be constructed from the virus, minimum activity, and idle power estimates if the amount of time spent in each condition is known. This greatly reduces the number of tests that detailed analysis corresponding to low levels of abstraction must be completed for. Instead architectural level of abstraction may be used to determine the percentages of time in each state $perctime_{(i)}$. While not shown, the leakage (a function of voltage and temperature) may also be bounded by the virus, TDP, idle and minimum activity test cases by finding the temperature under these work loads. Thus, for a scenario that spends time in n different states, the total average power for that scenario is estimated as

shown in equation B.6.

$$C_{eff(i)} = AR_{(i)} C_{eff\text{-}virus}$$

$$P_{active(i)} = C_{eff(i)} V^2_{(i)} F_{(i)}$$

$$P_{total\text{-}scenario} = \sum_{i=1}^{n} [(P_{active(i)} + P_{lkg(i)}) perctime_{(i)}] \qquad \text{(B.6)}$$

This is also significant because the virus, minimum activity and idle tests can generally be constructed with a small enough set of assembly code instructions so as to be executed on a functional RTL model in a reasonable amount of time. These tests are practical to use during the pre-silicon design phase to track power convergence and feed into the various power delivery system, thermal control system and power management system design processes. This leads naturally to a discussion regarding the different levels of abstraction employed for power estimation.

B.2 Levels of Abstraction for Power Estimation

B.2.1 Architecture

Architectural level of abstraction is usually the earliest phase of the design process. At this early stage in the design process effective capacitance C_{eff} and effective transistor width W_{eff} are used to do power trade-off studies. These studies enable comparison of various micro-architectural decisions that are being considered. Note that W_{eff} is used to estimate leakage. Using

a similar method (not shown) as C_{eff} is used for active power as shown in section B.1.

C_{eff} might represent a small array or an ALU in the context of architectural analysis of a CPU; alternatively, it might represent a large subsystem like a CPU or GPU in the context of analysis on a SOC. C_{eff} is a powerful method of representing the active power of a portion of the design because it allows a designer to think and scale C_{eff} up or down with different changes in process technology and/or increasing the size of the block corresponding to architectural design decisions. Notably though, the power may then be computed from the value of C_{eff} at various voltage and frequency points. Similar use of W_{eff} for tracking how leakage will scale, as different device types are used to trade frequency for leakage as a function of voltage and temperature. This enables an architecture team to make changes to the design updating C_{eff} and W_{eff} while other team members look at circuit, voltage and device type trade-offs that may be used to optimize the voltage, power and frequency targets for the design. This also gives the architecture team a proxy for both active and leakage power so that they are capable of making high level power vs performance decisions before the circuit optimization is completed.

B.2.2 RTL

Performance modeling teams are typically interested in correlating the high-level events that are used to predict performance between the behavioral model (generated during the architectural level of abstraction) and the func-

tional RTL. This correlation of event activities can also be used for power estimation using the power/event data from the architectural design phase. Later, once the power of the blocks can be accurately estimated (at the end of the gate-level analysis) the actual power per event can be modeled and employed in power management techniques (see chapter C) during real time operation of the silicon in a customer environment.

Furthermore the behavioral model, typically capable of running much faster than actual RTL simulations, can be used to find windows of high, typical and low activity in real traces end users will run on their systems. These can then be windowed and run through the RTL model to get lower level data on signal activities. This windowing work is important, and the RTL traces identified at this time will be later used on the gate-level design resulting in the final pre-silicon power estimates for each block in the design.

The earlier actual RTL (functional model) can be pushed through synthesis, the better. Even if a library (for the target process technology) is not available, an older process technology library may be used. This early synthesis (even on incomplete RTL) can test the initial features and start to give designers feedback on the area, capacitance and width of the design. This helps refine C_{eff} and W_{eff} values used by the architecture team. Even though the RTL quality may be low (bugs) and features not yet implemented, this work accelerates moving to the gate-level of hierarchy power analysis and gives the designer power convergence trends highly valuable to reducing power and tracking overall project progress.

B.2.3 Extracted - Gate and Layout

Gate-level analysis generally involves supplying a relatively small number of vectors from an RTL simulation to a gate-level netlist in order to link the explicitly functional behavior of the RTL to the specific capacitance (actual gates and wires) used during implementation. Formal equivalency checking already requires inputs, outputs and state elements be matched between the RTL and the gate level netlist. These matched nodes are used as the sync points to feed the RTL simulation switching and signal levels into the gate-level netlist. In this way, the activity factor and state probability of every node in the gate level netlist may be calculated.

The switching on every net is converted into an activity factor (AF) (as in equation A.2). Furthermore, the capacitance for each net is obtained from extracted layout. Summing equation A.2 across all nets in the design yields the active power. If the active power of the gates themselves was characterized and included in the library's power files, then short-circuit power may also be included in the estimation. Finally the power of the block (as predicted by the sum of all cells and nets - as a function of the specific capacitance and activity factors associated with them) should be compared back to the block's original C_{eff}. Similarly, leakage estimates directly from the circuits and specific vectors should be used to confirm that the leakage power matches that corresponding to the W_{eff} being used by the architecture team. In this way, the design team can ensure that the block does eventually reach the targets specified during the architectural design phase, or the architecture team needs to absorb new

circuit level data to ensure power predictions at all levels of hierarchy are self consistent.

B.3 Correlating Predictions with Measurements

For completeness, it is very important to make detailed measurements on a wide variety of material (voltages, temperatures and frequencies) to determine how close the pre-silicon estimates were to the actual results. Checking the various tests (virus, idle, minimum activity) in isolation will enable the team to determine if there are any power bugs in the device. Measurements of TDP and other interesting scenario power is critical to validate the performance goals can be achieved. These activities are also very important to further refine the C_{eff} and W_{eff} estimates for proliferation product development.

Appendix C

Power Management Methods

Power management refers to the process of both minimizing the power consumption required to achieve a certain target performance level and ensuring the device stays within the required operating ranges (voltage, temperature and frequency) for safe, reliable operation. Thus, changes need to be made by the power management control system in order to optimize the power consumption depending on the demanded performance level. However there are costs associated with making any change in the behavior of the system. Simplistically, these costs may be lumped in to three categories:

- Added Power: expended to achieve a lower power state for a long time
- Time: to make the change; response time of the system as viewed by the user and applications. Sometimes entry latency and exit latency (from a particular condition) may be considered separately
- Break even time: when going from a high performance state to a lower performance state (to save power) this is the amount of time required to stay in the new (lower) power state before returning to the previous (higher power) state in order to save more energy than it cost to make

the change. This is distinct from latency which is only the amount of time required to make the change.

Several papers describe different power states, associated latencies and power management mechanisms that are used[39] [54] [49] [22]. One of the most common power management techniques is Dynamic Voltage and Frequency Scaling (DVFS). Exploring it in detail serves to illustrate the DVFS method and the impact of the costs associated with using power management techniques.

C.1 Dynamic Voltage and Frequency Scaling (DVFS)

If the operating system determines that the processor has very low utilization but is operating at the maximum possible frequency then it may signal the processor to change to a lower frequency (Dynamic Frequency Scaling). In this way active power is reduced since it is proportional to frequency. The leakage stays the same unless the temperature reduction that comes with the active power reduction is included. However, it may be possible to reduce the voltage with the frequency as long as the processor is not already operating at the minimum voltage. In this case a DVFS transition is made. This greatly reduces the power consumption as compared to a simple DFS transition. As discussed previously leakage and active power are strong functions of voltage. It is notable that this significant reduction in power may cause enough drop in temperature that the voltage may later have to be bumped back up to account for the frequency dependencies on temperature. These different voltage and

frequency combinations are examples of processor states (p-states) described in more detail later.

C.1.1 DVFS Costs

To consider the costs associated with the DVFS event one must look at the individual steps required to make the change:

- halt execution of instructions: some power cost to flush the pipe-line and time or opportunity cost because code execution stopped
- signal the VR to increase/reduce the voltage: control signals switching that consume active power, also the power supply energy $\frac{1}{2}C_{supply}(\Delta V_{supply})^2$
- wait for the voltage change in the power delivery network to stabilize: cost is time that no work can be done (and leakage burned during this idle time)
- wait for the PLL to re-lock at the new frequency: more time lost, leakage and active power of the clocks during lock where no useful work can be done
- Finally perform a code fetch and refill the pipeline in order to execute instructions

It may take several micro-seconds of time (PLL lock + waiting for the voltage to stabilize). While the change in supply voltage is small, the capacitance on the supply network is generally very large relative to the effective

capacitance of the die under normal operation. In a pathological case one can envision a poorly designed power management system that ends up switching back and forth between different DVFS states consuming a lot of energy but not getting any work done. While a single DVFS event may not be noticed in real time by an end user, its possible for a poorly designed power management system to create several of these events per second resulting in performance losses and increased power consumption. This will be noticed over longer periods of execution. There are other hardware alternatives that have different DVFS overhead and costs. For example, a design could have two PLLs implemented so frequency does not change until the second PLL has locked. This saves time during the transition, but it also adds the power and area of a second PLL.

C.1.2 DVFS Thermal Considerations

Thermal response will lag changes in the power consumption. One way to resolve this is to leave the target voltage at each DVFS point high enough so that the corresponding frequency at each point can be achieved for all temperatures. However, this essentially means operating the processor at a voltage higher than required most of the time consuming extra power and leaving the product at a competitive disadvantage. A more robust power management system may comprehend the temperature impact on DVFS changes. If the power management system only reacts to the temperature then two more DVFS transitions will be made. One changes performance and the second ad-

justs for temperature. The change due to temperature occurs at a later time due to the higher time constant of the thermal response of the processor's environment. An alternative power management system attempts to predict the temperature change over time in advance and reduce the number of DVFS events. This method requires additional complexity and design effort. The extra complexity leads to additional circuits that consume more power; but the goal is to reduce power consumption. In order to determine if it makes sense to add circuits and power, to save more power under different work load scenarios, a solid understanding of the target work-loads and full-chip power consumption is required.

C.1.3 Summary: Savings, Costs, Break Even Time

The DVFS discussion above demonstrates that the power management mechanisms inextricably embody the following aspects:

- the power savings opportunity resulting from the change (using the power savings feature)
- the time cost associated with the change (entry and/or exit)
- the power cost associated with the change (data movement and/or power up/down capacitance)
- a break even time (time in the new state required to save as much energy as it cost to make the change before another change is required)

It is important to have many different performance and power states available - assuming these options gradually move from small to large power savings (with correspondingly small and large costs). It is equally important the power management system embody enough intelligence to effectively use those mechanisms. As the costs to use a power savings feature increase so do the risks of getting it wrong resulting in increased overall power consumption of the processor. The pre-silicon decisions to build in hardware support for a particular power savings feature implies that the intelligence required to effectively use that feature has also been well thought out. For example DVFS methods usually employ hysteresis, preventing too many consecutive changes, ensuring stability.

Extreme examples of on-die multi-processor systems with eight voltage islands and 28 frequency islands covering a total of 48 cores, using off-die voltage regulation, have been demonstrated in hardware[25]. Additional work is being done at the software level to optimize power when a specific total throughput work load constraint is applied to chip multi-processor systems having fine-grain DVFS capabilities[40].

To further motivate power management concepts of: power savings opportunity, power costs, time costs, and break-even time, consider another pervasive power reduction technique used today.

C.2 Clock Gating

Clock gating is a term used to describe the process of turning off the clocks to circuits that are not processing data. The clocks could be shut off to an adder when there are no adds in progress, but clock gating is just as easily applied to virtually all circuits (arrays, branch predictors, decoders, etc.) on modern processors. Clock gating is generally considered a power reduction technique as opposed to a power management technique because it reduces overall power - virus, TDP and minimum-active. Without clock gating most systems would require redesign in order to deliver increased power demanded by today's processors.

Clock gating is also vital because, by definition, clocks have an $AF = 1.0$ left free-running. Processor statistics indicate that most nets in a processor are idle or have extremely low activity factors[23]. Yet these processors may be consuming 100W of power or more. It is interesting to note that most power virus tests are an attempt to generate a code stream which ensures the large blocks on the processor (with a lot of clock power) are active. However, it is generally not possible to keep all blocks active at the same time. Some execution operations take several cycles to complete and waiting for them to finish stalls other portions of the processor. Going to last level cache for data slows down execution. Thus, in a very real sense, clock gating reduces the virus power in addition to functioning as a hardware power management system reducing power when certain units are not needed to meet performance requirements.

C.2.1 Local Clock Gating

At the level of an individual circuit the decision to clock gate is often easily determined by examining the costs and benefits:

- power cost: related to an enable signal that must be created, sending that signal to an AND gate in order to qualify the clock prior to distribution of that clock out to the sequential cells. As long as the amount of switching capacitance of the control circuit is low relative to the amount of capacitance on the clock network being gated there will be a net power savings (additional leakage of the extra control circuit gates is usually minimal)

- time cost: timing constraints are such that the enable must arrive a phase (half a cycle) before it's used. This is a very small amount of time and it also scales with processor frequency. As a result the latency impact to enter/exit the local clock gating state is very low.

- break even time: taken to extreme (a unique clock gating control circuit for every sequential) it's possible to add more power than is saved. But in general it is easy to prove when the power saved is always lower than the power cost - independent of the work load being run by the end user.

C.2.2 Global Clock Gating

While local clock gating occurs on a fine grain level, global clock gating is the process of turning off the entire global clock distribution to a large

portion of the processor or SOC. Thus the main processor is not capable of doing any work.

The power savings can be large. Global clock distribution power falls in the range of 1-10% or more of the total virus power. It varies dramatically with the particular implementation used.

The power cost to turn off the global clocks is quite small (again a small control circuit) that operates on a separate clock domain (that may not be shut down). It does imply that there is at least one other domain with an active clock so that the clocks can be turned back on at a later date and interrupts can be handled. There are implementation complexity costs associated with this and perhaps some overlap between multiple clock domains. There may also be increases in skew between the clock domains which in turn reduce the amount of usable cycle time available to signals that propagate between sequential cells on the two domains. This may cause power increases in those specific circuits to meet timing. There may be other costs associated with Design For Debug/Test/Manufacturing to get special circuits to work properly.

The time cost for global clock gating is generally related to the propagation delay of the clock signal through the GCDN network. This may be on the order of a 0.5-5nS; however some low power GCDN networks (such as resonance frequency methods) may take much more time to stabilize after start up. However, if the processor has truly stopped executing code, then it will remain that way for at least a few cycles. It's also important to note that the processor can not start executing code immediately. The operating

system needs to know that the global clocks are off in order to start the process of turning the global clock back on prior to executing any code. In fact, it's possible that software originally turned the global clock distribution off.

For most processors and SOCs global clock gating saves more power than it costs. Furthermore, the wake-up time is low enough as to be transparent to human users. Often the break-even time is just a few clock cycles - much faster than most real-time, external interrupts.

Nevertheless, clock gating, as a power reduction technique, is employed in two different ways with different cost trade-offs. Local clock gating is purely controlled by hardware and independent of any software control or power management policies. Global clock gating needs some interaction between the hardware and software for effective use.

C.2.3 Impact to Power Delivery

There is one more notable implication of clock gating on the power management system - it can lead to higher magnitude di/dt events and thus additional voltage guard band requirements. A pathological case, but events like this do happen, is where the processor's caches are preloaded with a power virus like test. However, the test is not executed right away either because of an interrupt or something else the processor must handle. If that interrupt is not doing very much work, the clock gating will turn off all the power associated with the circuits not in use. This greatly reduces the total power consumed by the processor. However, when the interrupt finishes and the high

power test that was loaded into the cache starts, the circuit blocks activate in the next clock cycle. In just a few clock cycles the execution pipeline fills up. This results in a very large change in active power, over a very short period of time introducing a high di/dt event. This leads to higher voltage in order to compensate for the voltage droop events induced by the di/dt events. The increased voltage required to maintain the frequency reduces some of the power savings achieved by the clock gating. Care must be given to ensure any new power savings methods are fully analyzed in order to determine if they create a new 'worst-case' scenario for some other dimension of the power consumption, delivery or thermal systems. If they do, then those costs should not be neglected during the trade-off analysis.

Some power management mechanisms can be implemented in pure hardware. In general, power management mechanisms that rely purely on hardware are intended to reduce power without impacting the system's performance. If system performance may be significantly impacted, then it is advisable to use some sort of software (firmware or other mechanisms transparent to the end user) so that the policies and operation can be tuned based upon post-silicon analysis. Other power management mechanisms directly leverage software control (from particular applications or from the operating system) to interact with hardware power management capabilities. Software may set policies that act as tuning parameters used to guide the hardware's behavior. Alternatively software may explicitly initiate power management state changes. For all cases proper analysis and post-silicon testing must be

completed to ensure the system is in fact reducing power - across all types of working environments, operating systems, and applications. Increasing the power or degrading the end-user experience will impact the success of the product in the market place.

C.3 Advanced Configuration and Power Interface (ACPI)

There is an industry standard Advanced Configuration and Power Interface (ACPI) Specification, which exists to establish industry common interfaces allowing robust operating system (OS)-directed motherboard device configuration. This specification may be found at the ACPI website: `http://www.acpi.info/`. ACPI is the key element in Operating System-directed configuration and Power Management (OSPM). It is important to note this specification is used for end-user electronics and other systems that are not general purpose computers.

C.3.1 ACPI State Definitions

The ACPI specification defines several types of state which are used to generically describe the relative power consumption and costs associated with entering/exiting/staying-in these states. For more specific details the ACPI specification should be consulted directly. Generally the state with the number zero is an active or working state. As the number increases the performance and power consumption are reduced but the energy and time costs to use those states increase.

- Global system states (Gx states) apply to the entire system and are visible to the user
  - G0 = working state
  - G1 = sleep state
  - G2 = soft-off state
  - G3 = mechanical-off state
- Device power states (Dx states) are states of particular devices are generally not visible to the user. For example, some devices may be in the off state even though the system as a whole is in the working state
- Sleeping states (Sx states) are types of sleeping states within the global sleeping state, G1
  - S0 = working state
  - S1 = sleeping with processor and DRAM context maintained
  - S2 = sleeping with processor context not maintained but DRAM context maintained
  - S3 = like S2 but additional HW may be turned off
  - S4 = sleeping with both processor and DRAM context not maintained
  - S5 = soft-off

- Processor power states (Cx states) are processor power consumption and thermal management states within the global working state, G0
  - C0 = working state executing instructions
  - C1 = no instructions are being executed (halt instruction was last to be executed), PLL and GCDN still active
  - C2 = some global clock gating may be employed but hardware responsible for keeping cache coherent among multiple system bus agents
  - C3 = additional global clock gating may be employed and OSPM assumes responsibility for maintaining cache coherency (flushing the caches in advance)
- device and Processor performance states (Px states) are power consumption and capability states within the active/executing states, C0 for processors and D0 for devices
  - P0 = maximum performance state possible
  - P1 a lower power and lower performance state than P0
  - Pn the lowest power and performance state possible while still doing work (where n is the maximum value in a given system)

P-states are often associated directly with DVFS operating points. However, it's possible to change the maximum power and performance that

can be consumed within a particular DVFS operating point. Thus, DVFS operating condition options may only represent a small portion of the actual number of P-states available in a processor. Furthermore, it's important to note the actual power consumed, in a given P-state, will still depend on the instructions being executed. Only the range of possible performance and power levels are modified by P-state changes.

"Enhanced Dynamic Acceleration technology"[54], "Turbo-boost"[24], "Burst-mode"[64] states have recently been introduced. The idea behind these modes is to run some small portion of the system (perhaps a single processor in a multi-processor system, or a single thread in a single processor) at a higher performance level than would otherwise be possible if all processors were active simultaneously. In some cases the reliability and thermal limits might not be reached (because only a portion of the processor is active). However it's likely that eventually either the power and/or thermal constraints will be exceeded. When the limit is reached the processor's voltage and frequency must be reduced to maintain long term reliability. It can be argued these higher performance states are the true P0 state of the processor; however the processor must typically (TDP) operate at some lower P-state condition due to power and/or thermal limitations.

- Thermal Throttling states (T states)

Transitions to T-states may be initiated by either hardware or software. From a hardware perspective, modern processors employ built in thermal sen-

sors which force a processor to either transition to another state in order to prevent overheating or circuit damage. These may be lower P-states where performance is reduced but the system continues to function. If the temperature rises too quickly and crosses a limit the mechanisms can force an immediate shutdown (for example to G2-state). Software may also initiate throttling state changes based on other sensors placed in the system. An example is when the exterior of a hand-held device gets too hot the OSPM initiate a change to a lower P-state preventing it from getting worse. Figures C.1 and C.2 show the ACPI G-state and C-state relationships respectively. These were taken from the ACPI spec 4.0a found at `http://www.acpi.info/`.

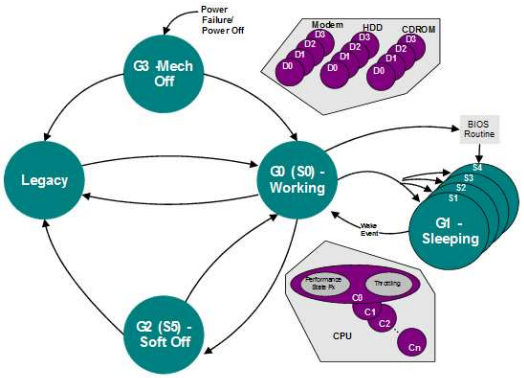

Figure C.1: ACPI G-state relationships

C.3.2 Processor C-states in More Detail and Their Power Management Trade-offs

C1, C2 and C3 were briefly described in section C.3.1. The supply voltage is not changed when transitioning from C0 to any of these C-states.

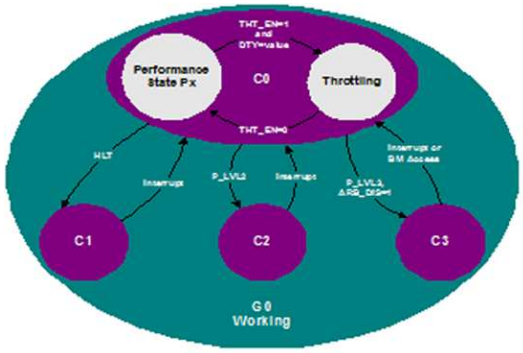

Figure C.2: ACPI C-state and P-state relationships

The costs associated with C2 (global clock gating) were discussed in section C.2.2 as an example of how power management systems work and the costs associated with them. C3 takes clock gating a step further by shutting down the PLLs and turning off some of the other clocks still active during C2. The definition of C3 is that the OSPM assumes responsibility for maintaining cache coherency with other processors and agents on the system bus. Thus, C3 saves additional active power (PLL and other clocks) but it requires more time for the PLL to lock when returning from C3 to C0. The costs are the re-lock time of the PLL (generally 1uS - 5uS) and the power consumed during the re-lock process when no work is accomplished. Other C-states and S0i*-states have been defined and used as well as new features and interactions between OSPM and hardware power management[3]. A few additional states will be introduced to further illustrate the trade-offs of power to enter/exit the states, the possible power savings, the latencies to make the transitions and the break-even time.

C4 is very similar to C3 but the voltage is dropped to the minimum level required to retain state. Retention voltage is lower than the minimum voltage required during normal operation, when memory elements in arrays and sequential cells must experience read/write operations at speed. Portions of the L2 cache may be flushed and completely disabled in this mode.

- Power cost to enter/exit C4 is the $C_{supply}(\Delta V)^2$ on the power delivery system's capacitance when idle. There are additional power costs if certain portions of the L2 cache are flushed and completely shut down
- Time penalty is similar to C3 but additional time to bump the voltage up and down (and stabilize) is required. There is additional time penalty if certain ways of the L2 cache are flushed prior to entry
- Power savings is the leakage difference between being in C3 and C4 which is relatively small depending entirely on the difference between the retention min voltage and the active min voltage levels

C6[22] (or Deep Power Down)[54] is a state where the processor's context is read and moved to a separate SRAM so that the power to the processor can be turned off (or reduced below the min voltage required to retain state). All caches (L1 and L2) are flushed prior to C6 entry. When the OS signals the processor to wake up, the processor context must be reloaded from the SRAM where it was stored prior to the start of the execution of new instructions.

- Power costs to enter and exit C6 are again related to $C_{supply}(\Delta V)^2$ on the power delivery system's capacitance when idle. Also, the energy to store the processor's context (data movement) may be significant especially when a large quantity of data in the L1 and L2 must be flushed. The target SRAM for the processor's state must be powered up (if it's a dedicated SRAM) or flushed (if a portion of another system SRAM is being used which could impact system performance elsewhere).

- C6 transition times are much larger than C4 - possibly 100s of micro seconds[22]. Much more data is moved, taking more time. The change in voltage on the power delivery system is much larger in order to increase the power savings.

- Break-even times for C6 may be quite large and are estimated by comparing the power difference from the C4 state to the C6 state to the active power consumed just by powering the voltage supply down and up again $C_{supply}(\Delta V)^2$. These break even times may easily be larger than interrupt rates that certain applications will demand (video and/or audio playback where streaming data is involved). The OSPM must ensure C6 is only used when the interrupt rate demands of the applications are low enough to net an overall power reduction without impacting the end-user experience.

The S0i1 and S0i3 states refer to situations where the processor core is off or in the C6-state and other components in the SoC are enabled or disabled

to facilitate different responsiveness and performance as demanded by the end user[3]. On die power-gating is used to save power for certain blocks that are not in use and has several advantages.

C.4 Power Gating

To help reduce the break-even time for low power states design teams have to reduce the power delivery capacitance that experiences the transition by using power-gating, sometimes referred to as a sleep transistor - instead of signaling the VR to change the voltage. Some processors may have 19 or more clock and power-gating islands[2]. This has the benefit of speeding up the transition time which can be directly controlled on die, avoiding a communication protocol to the VR. The portion of the power delivery capacitance that experiences the transition (to the capacitance after the power-gate structure) can be cut by a factor of 10 or more. This not only saves a lot of power associated with the transition, but it also speeds up the settling time of the power delivery network when exiting the low power state. The introduction of a series power-gate in the power delivery network also increases the series resistance and inductance between the transistors and the package and/or the motherboard. This introduces additional IR power loss through the resistance, another power cost, and it also introduces additional sensitivity to di/dt events. The same magnitude of di/dt event may now induce a larger voltage droop and thus require more voltage guard band. Very high current processors, using power-gating, may reconnect the die-level power delivery network up to the

package in order to add package decoupling capacitance[50] [11]. In order to truly optimize the hardware for multiple power and clock domains, designers must comprehend the work-loads that are important to their customers[20].

Once stabilized in the low power state the power-gate method does not turn off the voltage regulator which is still active and generating a leakage current - very inefficiently. The absolute power savings of the power-gate method, as viewed purely in the steady state condition, may not be as large at the system level as the method which turns off the VR directly. However, the significant reduction in break-even time means that it is much more useful across many work load conditions and may be used more effectively by the OSPM.

It is critical to realize that, there is still a significant difference between the power-gate method and the VR controlled method. Specifically, the power-gate introduces additional power costs that impact the processor during it's normal operation. The additional IR losses, and likely the additional voltage guard band, resulting from increased series resistance of the power-gate means that the power virus condition has become worse than it would have been if the VR method had been used to change the voltage associated with the low power state change. Using the VR to turn off the power supply has larger entry/exit costs, but they're only paid when the OSPM directs the hardware to use the state. The power gate method reduces the costs of both power and time associated with each entry and exit of the state, but it introduces a power cost that must be paid even if the OSPM never uses that low power state.

Knowledge of how often the OSPM will be able to use one implementation vs another is critical to include in the break-even analysis. End-user work loads today have enough range that the solution could go either way depending on the assumptions the design team makes about the applications they believe the end user will use more often. The worst-case virus and TDP applications now all consume more power; is this acceptable? It can be challenging to ensure all of the costs associated with a particular power management technique are properly captured and analyzed.

C.5 Voltage Guardband Reduction

C.5.1 Voltage Droop Detection

For years failure analysis and debug engineering teams have used mechanical micro-probing, more recently TRE[21], to look at the power supply on silicon during a test to determine what the transient response of the power delivery system looks like. This is used extensively to determine if transient current events are causing the voltages to droop more than expected. More recently teams have developed analog[9] and digital[44] voltage droop detection circuits that do not require physical probing. A ring oscillator's frequency will change with the voltage allowing higher voltages leading to faster frequencies. The frequency of the ring oscillator should be much higher than the fastest clock in the processor being tested. For example, suppose the ring-oscillator is running ten times faster than the processors clock at the target voltage. If the counter was read and then reset after the rising edge of every processor's

core clock then the counter should read ten every cycle. If the counter read resulted in a value of nine, instead of ten, it would indicate that a voltage droop occurred which slowed the ring oscillator down, assuming temperature and other variables were controlled and constant. If the counter was read and then reset after every ten processor clock cycles, then the counter should have a value of 100 if the average voltage over the ten core cycles was equal to the target voltage. It's possible that the voltage was lower and the ring oscillator slower for some of the 10 cycles and then faster for the remaining cycles such that the count was still 100. In this case the per cycle data would have been lost, but the ring oscillator's sensitivity has increased because now an average voltage change resulting in a 1/100 difference can be detected instead of 1/10 with the per cycle measurement. Post silicon debug teams have been able to use these methods to compare the per cycle power delivery voltage with the cycles that they have found speed-paths using clock-shrinking and or other methods. This can give a good indication of situations where di/dt events are inducing speed-paths in the silicon because there is not a large enough voltage guard band.

C.5.2 Voltage Droop Induction

Transistors may also be added and used as voltage droop inducers[45]. Using other design for test capabilities found on micro-processors, these transistors, when enabled, act as a short between the supply and ground rails on the silicon. They may be turned on for one or more cycles and the number of

cycles that they are on may be varied. Thus, di/dt events of various magnitudes and time durations may be created and applied at specific times during testing. This may be used extensively prior to production to find circuits that are particularly sensitive to di/dt events. For example, suppose there is a speed-path that's occurring when the voltage droop is only half as bad as it might be under different conditions. The team could use the droop inducing circuits to pull the voltage down to the worst case at the moment the speed-path is occurring in the test so it is possible to observe how much more the frequency might degrade. This is important because it might be months before that specific scenario can be identified in a system. This gives the post-silicon debug team the ability to predict how much voltage (and frequency) guard bands should be applied during manufacturing.

C.5.3 Reducing Guardbands

Thus far, the use of droop detect and droop inducing circuits has been illustrated as a debug tool used only by manufacturing teams to perform analysis prior to shipping the product. However it has been proposed that various circuits and methods be used to deal with these voltage droop situations and other issues such as process variation, transistor degradation over time, etc. Some propose the use of sensitivity circuits to detect real-time voltage, temperature or reliability degradation issues that may cause speed path failures which they compensate for through adaptive body bias, DVFS or other methods[33] [27] [8]. It is important for these circuits to be physically placed in

the locations that are most likely to be sensitive to the di/dt events which will vary depending on the RC nature of the local power delivery grid on the silicon. Still other methods have been proposed to use fault tolerant design methods to detect failures induced by dynamic real-time variation events and then re-play the failing operations at lower frequency to ensure they work properly[31] [46] [47]. This does have the unfortunate side effect of adding clock power as all the sequential cells in the active block were just duplicated.

A hybrid approach, merging fault-tolerant circuit methods with specific stand-alone sensitivity circuits has also been evaluated[15]. The fundamental goal of all these schemes is to reduce the required voltage guard band so that the processor can achieve the same frequency using a lower voltage supply saving power. While power should go down in a super-linear way, with the voltage reduction, care must be taken with methods where sequential cells are added and clock power increased. The increased clock power, and at-speed detection and control circuits, will eat away some of the power savings. More insidiously, the di/dt events may actually be made worse - as the effective capacitance switching increases. This will further reduce the savings of the technique. Finally, adding a significant number of state elements, relative to the total state elements in the processor, can impact the overall minimum voltage and limit the DVFS opportunity at the low end. Generally the latency of these systems is relatively low. In some of the cases the voltage is not changed, the operation is just re-executed. Small changes in the voltage supply may be accomplished on the fly without stalling the processor and waiting for

it to stabilize[34]. Most of the trade-off analysis results in a power calculation contrasting the cost of the implementation with the derived savings.

It's notable that most processors have a PLL which obtains power from an isolated power supply - in order to keep jitter low and PLL performance high. Often the global clock distribution is powered by the same digital voltage rail that the core processor's circuits are using. While there are variants on this, the global clock distribution and digital circuits use a voltage supply rail that is isolated from the PLL voltage supply, to protect the analog circuit performance of the PLL. While the digital circuits experience frequency shifts with the ups and downs of the digital voltage supply, the PLL generally continues to turn out a clock signal with exactly the same frequencies. If the global clock distribution is on the same voltage rail as the digital circuits, then there will be edge-placement movement - usually accounted for by skew. If the clock at the sequential cells could speed up and slow down along with the digital circuits (in sync) then the skew tax could be reduced and the frequency improved at a given operating voltage. When a local voltage change is detected in this implementation[42] [53], the local clock automatically speeds up or slows down to stay in sync with the digital signals going between the flip-flops receiving that clock. The local clock frequency is updated within one cycle and remaining clock domains are updated within two cycles. These are spatially correlated so that the clock frequency applied to the sequential cells is correlated to the change in data signal propagation delay between them.

These are examples of a pure hardware power management methods

used to reduce the impact of di/dt events and recover some of the voltage and frequency guard bands. The time costs associated with this particular implementation are very low - because the hardware detects the problem and passes the slow-down or speed-up signal along to the other clock domains at the same rate of propagation that the data signals are moving, either slightly slower or faster. The number of clock cycles that a given domain stays slow or fast is minimized. So the power costs of circuits needed to reduce the guardbands must be lower than the power saved by being able to run at a reduced guard band. Some of these systems introduce stability issues[43] that must be analyzed and prevented.

C.6 Predictive Power Management Methods

C.6.1 Correlation of Significant Events to Power

Another interesting power management technique is to correlate processor events with the power consumption of the processor[19]. Some examples of events that might be useful for power estimation and correlation are:

- the execution of different instructions may have different power weightings
  - a double precision SIMD instruction will probably consume more power than a simple single precision instruction - even though the absolute power associated with either will depend on the actual data values (and data path capacitance toggling) which would not

be known in advance.

  - floating-point and integer instructions executed - these two different types of arithmetic will consume different amounts of power on average.

- number of instructions decoded, issued, retired.
- number of loads, stores, cache miss, cache hit, predicted branches, mis-predicted branches.
- different blocks may have a power associated with their clock gate enable signals. These are signals controlling the clocks to large circuits, with many sequential cells weighted more than signals controlling a smaller number of sequential cells.

C.6.2 Improves Early Power Estimation

Correlation of these statistics to power consumption is highly valuable. The data can be leveraged during the pre-silicon design phase to correlate events monitored in the performance simulator with power predicted by gate level simulators. This is very important because the gate level simulators in use today are limited to simulating a few thousand processor cycles through RTL - thus only small time slices of real applications can be modeled at the gate level. Performance simulators are capable of running several million processor cycles. Entire benchmarks and significant portions of real code, including the OS, can be simulated by the performance model. If a reasonable correlation of

gate level power can be linked to performance level events, then accurate pre-silicon predictions of the power consumption for interesting work-loads may be generated. This will result in several benefits:

- quantitative data which may then be used to help make power management decisions between virus, TDP, average, minimum-activity and idle scenarios.

- software (OSPM) and hardware interactions can be modeled and developed well before silicon arrives. As software evolves to improve performance the frequencies will increase and the spacing between OSPM events will reduce. This tends to bring out additional droop and speed-path interactions which will need corrective actions via OSPM adjustments or additional voltage or frequency guard banding (which costs power).

C.6.3 Enables Real Time Power Prediction On Die

There is an opportunity to use silicon events with signals and performance monitors which have been correlated to actual power consumption as a leading indicator of what will happen in the future[29] [32]. This method could be significantly better than relying on voltage droop and thermal detectors which are lagging indicators of system power. It has been proposed that predictive methods could be used to turn on droop inducer circuits in advance of large di/dt events. Then, when the actual power consumption of

the processor increases the droop inducers would be turned off. In this way the processor could effectively request more current (through the power delivery system) in advance of actually needing it - increasing the time (dt) over which a change in current (di) occurs. The would reduce the magnitude of the di/dt event.

An alternative method of reducing the impact of a predicted di/dt event is to slow down the rate at which instructions are executed. This would allow the processor's performance and power consumption to increase more slowly by throttling the micro-architecture for a number of cycles. This has the advantage of consuming less power at the expense of system performance. A similar method could be used for a processor going from a high work load and a high current level to a lower level of current consumption, preventing voltage overshoots which could impact transistor reliability over time. This would enable a processor to run at a voltages closer to the processes maximum improving performance.

Another opportunity for predictive methods is to try and avoid DVFS events triggered by an overheating event. Again, the processor could slow the rate at which it issues and retires instructions so as to keep the power (and thus the temperature) from rising above the threshold in the first place. Essentially shifting to a lower P-state but without the overhead associated with DVFS. If the temperature stops increasing and starts recovering, then the power management system could bump the P-state back up. This is significant because the costs of micro-architectural throttling are minimal. The latency

is also very low and it can be accomplished in a couple of core clock cycles. In the presence of good clock gating the power savings is immediately realized. Performing a DVFS change of P-state may be more efficient because the power will decrease with the frequency and voltage squared, but only if the new P-state will be maintained for a long period of time. The goal, is to have sufficient intelligence in the hardware to know when to throttle the micro-architecture performance for a short period of time vs when to do something more drastic such as a DVFS transition. Having both predictive and reactive inputs to a power management controller is very valuable.

It is important to note that the thermal system response time constant and the power delivery system response time constant are different. The power management system is essentially a closed loop feedback control system. The power management policies and mechanisms must be analyzed as such. It is advisable to have methods to tune both the hardware and software such that any instabilities found in post-silicon can be eliminated without re-spinning the hardware. It may be tempting to simply put enough hysteresis in to the power management system to ensure stability - this is equivalent to just adding extra guard band voltage and frequency guard bands to the products specifications. Doing this will result in a less competitive product as compared to a design which really has an elegant power management system that works well. However, a non-trivial analysis may result when the effort to design a robust power management system is compared with the benefits of minimizing Time To Market and Time To Volume with higher guardbands. The particular

skills found on a given design team may influence these design decisions and, in fact, may be the deciding factor.

C.7 Dynamic Reliability Management

There is additional research related to Dynamic Reliability Management (DRM) and/or Dynamic Temperature Management (DTM). While DTM methods tend focus on ensuring a temperature limit is not exceeded, DRM reacts to the current reliability state and then initiates DVFS events to boost performance or recover unacceptable reliability losses[14]. Others are exploring how to migrate work-loads by moving active threads between different processors on a SoC to prevent excessive hot spot generation in a particular area and thus enable higher performance[36]. This has been extended to 3-D multiprocessor designs as well - where the algorithm targeted higher power (hotter) applications on the processor closest to the heat-sink and lower power (cooler) applications on the processors farther from the heat-sink[52] [57]. These ideas introduce interesting and complicated power management issues. For example it may cost a lot of power to migrate a thread from one processor to another, however in a thermally constrained multi processor environment there may be a net throughput improvement if some of the threads occasionally land on otherwise lightly loaded cores[18]. There are several variations possible; one example might be to spread heavy-work loads among the most distant cores to reduce the thermal interaction during execution. However, if the code streams have strong overlaps on the same data address then a lot of additional power

might be consumed to maintain cache coherency. One can envision pathological cases for either extreme. The fundamental point is that, the cost analysis can become quite complicated and difficult to generalize through hardware alone.

Bibliography

[1] Freescale's leading-edge pmic solution designed for intel atom z6xx mobile platform. http://www.freescale.com/files/analog/doc/fact_sheet/PMIC INTEL1FS.pdf, 2010.

[2] Intel fact sheet - introducting the next-generation intel atom processor-based platform (intel atom processor z6xx series and platform controller hub mp20). http://www.download.intel.com/pressroom/kits/atom/z6xx /pdf/Fact_Sheet_Intel_Atom_Processor_Platform.pdf, 2010.

[3] AnandTech. Intel unveils moorestown and the atom z600 series the fastest smart phone processor. http://www.anandtech.com/show/3696/intel-unveils-moorestown-and-the-atom-z600-series-the-fastest-smartphone-processor/1, 2010.

[4] AnandTech. Motorola droid bionic - a quick preview. http://www.anandtech.com/show/4761/motorola-droid-bionic-preview.pdf, 2010.

[5] Anand Chandrasekher. Intel ® 2009 investor meeting. http://www.umpcportal.com/downloads/2009_05_IM_Chandrasekher_final.pdf, May 2009.

[6] Android Community. Droid 3 battery life. http://community.vzw.com/t5/Android-Discussions/Droid-3-Battery-Life-issues/td-p/645655, 2011.

[7] Zoli Erdos. Fixing the battery problems your android smartphone seems to have. http://www.enterpriseirregulars.com/19162/fixing-the-battery-problem-your-android-smart phone-seems-to-have, 2010.

[8] A. Drake et al. A distributed critical-path timing monitor for a 65nm high-performance microprocessor. *International Solid-State Circuits Conference (ISSCC) Technical Digest of Papers*, pages 398–399, 2007.

[9] A. Muhtaroglu et al. On-die droop detector for analog sensing of power supply noise. *Symposium on VLSI Circuits Digest of Technical Papers*, pages 193–196, 2003.

[10] Arun Chandrasekhar et al. Chip-package-board co-design of 45nm 8-core enterprise xeon processor. *Electronic Components and Technology Conference (ECTC)*, pages 536–542, 2010.

[11] Arun Chandrasekhar et al. Chip-package-board co-design of a 45nm 8-core enterprise xeon processor. *Electronic Components and Technology Conference (ECTC)*, pages 536–542, 2010.

[12] Bing Dang et al. Integrated microfluidic cooling and interconnects for 2d and 3d chips. *IEEE Transactions on Advanced Packaging Vol. 33, Issue 1*, pages 79–87, 2010.

[13] C. Prasad et al. Reliability studies on a 45nm low power system-on-chip (soc) dual gate oxide high-k/metal gate (dg hk+mg) technology. *International Reliability Physics Symposium (IRPS)*, pages 293–298, 2010.

[14] Cheng Zhuo et al. Process variation and temperature-aware reliability management. *Design Automation and Test in Europe Conference*, pages 580–585, 2010.

[15] D. Bull et al. A power-efficient 32b arm isa processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation. *International Solid-State Circuits Conference Digest of Technical Papers*, pages 284–285, 2010.

[16] Daniel Gmach et al. Profiling sustainability of data centers. *International Symposium on Sustainable Systems and Technology (ISSST)*, pages 1–6, 2010.

[17] Dhu-Hsing Lin et al. Energy consumption analysis of audio applications on mobile handheld devices. *IEEE Region 10 Conference*, pages 1–4, 2007.

[18] Dongkeun Oh et al. The compatibility analysis of thread migration and dvfs in multi-core processor. *International Symposium on Quality Electronic Design (ISQED)*, pages 14–17, 2010.

[19] Dongkeun Oh et al. Runtime temperature-based power estimation for optimizing throughput of thermal-constrained multi-core processors. *Design Automation Conference Asia and South Pacific (ASP-DAC*, pages 593–599, 2010.

[20] Efraim Rotem et al. Multiple clock and voltage domains for chip multi processors. *ACM International Symposium on Microarchitecture*, pages 459–468, 2009.

[21] F. Stellari et al. On-chip power supply noise measurement using time resolved emission (tre) waveforms of light emission from off-state leakage current (leoslc). *International Text Conference*, pages 1–10, 2009.

[22] G. Gerosa et al. A sub 2w low power ia processor for mobile internet devices in 45nm hi-k metal gate cmom. *Asian Solid-State Circuits Conference*, pages 17–20, 2008.

[23] Hans Yeager et al. Microprocessor power optimization through multiple device insertion. *Symposium on Very Large Scale Integrated Circuits Digest of Technical Papers*, pages 334–337, 2004.

[24] J. Charles et al. Evaluation of the intel core i7 turbo boost feature. *International Symposium on Workload Characterization*, pages 188–197, 2009.

[25] J. Howard et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 108–109, 2010.

[26] J. Tschanz et al. Adaptive circuit techniques to minimize variation impacts on microprocessor performance and power. *International Symposium on Circuits and Systems, Vol. 1*, pages 9–12, 2005.

[27] J. Tschanz et al. Adaptive frequenty and biasing techniques for tolerance to dynamic temperature-voltage variations and aging. *International Solid-State Circuits Conference (ISSCC)*, pages 292–293, 604, 2007.

[28] Jian Sun et al. Fully monolithic cellular buck converter design for 3-d power deliver. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 17, Issue 3*, pages 447–451, 2009.

[29] Jong Sung Lee et al. Predictive temperature-aware dvfs. *IEEE Transactions on Computers, Vol. 59, Issue 1*, pages 127–133, 2010.

[30] JW Tschanz et al. Adapting body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits, Vol. 37, Issue 11*, pages 1396–1402, 2002.

[31] K.A. Bowman et al. Energy-efficient and mestastability-immune resilient circuits for dynamic variation tolerance. *IEEE Journal of Solid-State Circuits, Vol. 44, Issue 1*, pages 49–63, 2009.

[32] Kishor Kumar Pusukuri et al. A methodology for developing simple and robust power models using performance monitoring events. *WIOSCA*, 2009.

[33] Kunhyuk Kang et al. On-chip variability sensor using phase-locked loop for detecting and correcting parametric timing failures. *IEEE Transac-*

tions on Very Large Scale Integration (VLSI), Vol. 18, Issue 2, pages 270–280, 2010.

[34] Lee Jeabin et al. A power management unit with continuous co-locking of clock frequency and supply voltage for dynamic voltage and frequency scaling. *International Symposium on Circuits and Systems*, pages 2112–2115, 2007.

[35] L.S. Sraboni et al. Improved vlsi circuit performance using localized power decoupling. *International Conference on Electrical and Computer Engineering (ICECE)*, pages 1–6, 2008.

[36] M. Cho et al. Proactive power migration to reduce maximum value and spatiotemporal non-uniformity of on-chip temperature distribution in homogeneous many-core processors. *Semiconductor Thermal Measurement and Management Symposium*, pages 180–186, 2010.

[37] M. Popovich et al. Effective radii of on-chip decoupling capacitors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 16, Issue 7*, pages 894–907, 2008.

[38] M. Suryakumar et al. Power delivery validation methodology and analysis for network processors. *IEEE Electronics and Technology Conference*, pages 589–592, 2004.

[39] M. Ware et al. Architecting for power management: The ibm power7

approach. *International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–11, 2010.

[40] Mohammad Ghasermazar et al. Minimizing the power consumption of a chip multiprocessor under average throughput constraint. *11th International Symposium on Quality Electronic Design*, pages 362–371, 2010.

[41] Nho Hyunwoo et al. A 32nm high-k gate sram with adaptive dynamic stability enhancement for low-voltage operation. *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 346–347, 2010.

[42] R. McGowen et al. Power and temperature control on a 90-nm itanium family processor. *Journal of Solid-State Circuits, Vol 41, Issue 1*, pages 229–237, 2006.

[43] R. McGowen et al. Power and temperature control on a 90-nm itanium family processor. *Journal of Solid-State Circuits, Vol 41, Issue 1*, pages 229–237, 2006.

[44] R. Petersen et al. Voltage transient detection and induction for debug and test. *International Test Conference (ITC)*, pages 1–10, 2009.

[45] R Petersen et al. Voltage transient detection and induction for debug and test. *International Test Conference (ITC)*, pages 1–10, 2009.

[46] S. Das et al. A self-tuning dvs processor using delay-error detection and correction. *IEEE Journal of Solid-State Circuits, Vol. 41, Issue 4*, pages 792–804, 2006.

[47] S. Das et al. Razorii: In situ error detection and correction for pvt and ser tolerance. *IEEE Journal of Solid-State Circuits, Vol. 44, Issue 1*, pages 32–48, 2009.

[48] S. Rusu et al. Power reduction techniques for an 8-core xeon processor. *Proceedings of ESSCIRC*, pages 340–343, 2009.

[49] S. Rusu et al. Power reduction techniques for an 8-core xeon processor. *Proceedings of ESSCIRC*, pages 340–343, 2009.

[50] S. Rusue et al. Power reduction techniques for an 8-core xeon processor. *Proceedings of ESSCIRC*, pages 340–343, 2009.

[51] Sangwoo Pae et al. Reliability characterization of 32nm high-k and metal-gate logic transistor technology. *International Reliability Physics Symposium (IRPS)*, pages 287–292, 2010.

[52] Shaobo Liu et al. Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor. *International Symposium on Quality Electronic Design (ISQED)*, pages 390–398, 2010.

[53] T. Fischer et al. A 90-nm variable frequency clock system for a power-managed itanium architecture processor. *Journal of Solid-State Circuits, Vol. 41, Issue 1*, pages 218–228, 2006.

[54] Varghese George et al. Penryn: 45-nm next generation intel core 2 processor. *Asian Solid-State Circuits Conference*, pages 14–17, 2007.

[55] W.A. Samaras et al. The ia-64 itanium processor cartridge. *IEEE Micro, Vol. 21, Issue 1*, pages 92–89, 2001.

[56] Wei Huang et al. Interactions of scaling trends in processor architecture and cooling. *Semiconductor Thermal Measurement and Management Symposium*, pages 198–204, 2010.

[57] Xiuyi Zhou et al. Thermal-aware task scheduling for 3d multicore processors. *IEEE Transactions on Parallel and Distributed Systems, Vol. 21, Issue 1*, pages 60–71, 2010.

[58] F. Arnaud et al F. Arnaud et al. 32nm general purpose bulk cmos technology for high performance applications at low voltage. *International Electron Devices Meeting (IEDM)*, pages 1–4, 2008.

[59] X. Chen et al X. Chen et al. A cost effective 32nm high-k/metal gate cmos technology for low power applications with single-metal/gate-first process. *Symposium on VLSI Technology*, pages 88–89, 2008.

[60] Roy Kaushik and Sharat C. Prasad. *Low-Power CMOS VLSI Circuit Design*. Wiley Interscience, 2000.

[61] Kelin J. Kuhn. Moore's law past 32nm: Future challenges in device scaling. *International Workshop on Computations Electronics (IWCE)*, pages 1–6, 2009.

[62] R. Shorten Rade Stanojevic. Distributed dynamic speed scaling. *Proceedings INFOCOM*, pages 1–5, 2010.

[63] Glenn Hinton Rajesh Kumar. A family of 45nm ia processors. *International Solid-State Circuits Conference (ISSCC)*, pages 58–59, 2009.

[64] Tomshardware. Processor power. http://www.tomshardware.com/reviews/intel-atom-moorestown-smart phone,2624-6.html, 2010.

[65] Neil H.E. Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective 3rd Ed.* Addison Wesley, 2005.